Polynomial Graph Commitments: Constant-Size Proofs for Directed Graphs with Cycles

Mischa Douwes
Independent Researcher
Netherlands
mischadouwes@hotmail.com

ABSTRACT

We present *Polynomial Graph Commitments* (PGC), a commitment scheme for directed graphs that instantiates bilinear polynomial accumulators [1–3] over the edge set. PGC maps a directed graph $G = (V, E)$ to an accumulator polynomial $P(x) = \prod_{e \in E}(x - H(e))$ over the scalar field of BLS12-381, where H is a domain-separated Blake3 hash preserving edge direction, and yields constant-size proofs of edge existence, edge non-existence, and batch membership—including over cyclic topologies where Merkle-based approaches incur $O(\log N)$ proof costs. Our contributions are: (i) a formal security analysis (Theorems 5.5–5.10) reducing membership, non-membership, batch, endpoint, temporal, and hiding soundness to q-SDH and Blake3 collision resistance; (ii) a fully stateless label-mode verifier via content-addressed edge hashing that removes the position map from the verifier's trust base; (iii) Merkle-chained epoch snapshots with domain-separated leaf/node/pad hashes and explicit setup-fingerprint binding, giving verifiable temporal proofs independent of graph size; (iv) a single tagged 81-byte wire format for existence and non-existence proofs, plus 96-byte BDFG21 batch witnesses; and (v) a production Rust implementation with a bounded 3-tier proof cache and a reproducible evidence bundle. Empirical benchmarks at 100–1,000,000 edges show median verification latency in the 3.9–4.9 ms band, size-independent, with a constant 81-byte proof payload (114 B with hiding). All claims are verified by a reproducible test suite (16 paper-level verification tests, 20 library tests, and the extended-scale benchmark, all passing in release mode).

CCS CONCEPTS

• **Security and privacy** → **Public key (asymmetric) techniques**; **Cryptography**; *Privacy-preserving protocols.*

KEYWORDS

Polynomial commitments, KZG, BDFG21, bilinear accumulators, graph commitments, non-membership proofs, q-SDH, BLS12-381, GDPR Article 17

1 INTRODUCTION

Graph databases model access-control policies, financial networks, supply-chain relationships, and identity graphs. A recurring need is to *prove* graph properties to third-party verifiers—that a relationship exists, or critically, that it *does not* exist.

Merkle trees are fundamentally tree-structured: committing a graph requires flattening it into adjacency lists, incurring $O(\log N)$ proof size per edge and no efficient non-existence proofs without Sparse Merkle Trees.

We observe that a directed graph's edge set is a *set of ordered pairs*. Mapping each pair to a unique field element and committing via a polynomial whose *roots* are these elements—a bilinear polynomial accumulator—reduces edge existence to a standard $O(1)$ KZG opening proof.

Contributions.

(1) **Formal security analysis.** We define adversary experiments for edge-membership, non-membership, batch, endpoint, and temporal soundness, and prove (Theorems 5.5–5.10) that PGC satisfies each under q-SDH and Blake3 collision resistance, via tight reductions to the KZG and BDFG21 opening relations.
(2) **Stateless label-mode verifier.** PGC-Label (§5.3) removes the position-map dependency: the verifier needs only the commitment, the SRS, the 81-byte proof payload, and the endpoint labels as external query context. Endpoint binding is content-addressed.
(3) **Temporal chain.** Merkle-chained epoch snapshots with three distinct Blake3 domain separators (`PGC_EPOCH_LEAF`, `PGC_EPOCH_NODE`, `PGC_EPOCH_PAD`) and an explicit 32-byte setup fingerprint binding each snapshot to the exact ceremony used (Theorem 5.9).
(4) **Tagged wire format.** A single 81-byte payload ($b \parallel v \parallel \pi$) distinguishes existence from non-existence on the wire, with constant size across arbitrary directed topologies including cycles; BDFG21 batches reuse a 96-byte witness independent of batch size.
(5) **Production implementation and reproducible evaluation.** A Rust implementation (≈ 3,500 lines, `src/pgc.rs`) with a bounded 3-tier proof cache, checked `try_commit()` API, and atomic epoch-log on-disk format. Benchmarks at 100–1,000,000 edges confirm size-independent verification and constant 81-byte proofs; the evidence bundle is regenerated by a single script and versioned under the repository's `verify/reports/` tree.

2 RELATED WORK

Graph Signature Schemes. Groß [7, 8] introduced graph signature schemes enabling zero-knowledge proofs of committed topologies under RSA. Tan *et al.* [9] proposed a q-SDH-based graph signature scheme for relational credentials. Yoshioka, Nakanishi, and Kitasuka [10] extend this line to zero-knowledge connectivity proofs for labeled directed graphs using a bilinear-map accumulator, with proof size and verification independent of graph size. These are *signature/credential* and topology-proof systems; PGC is a lower-level *commitment scheme* for edge membership, non-membership, and batch openings.

Bilinear Polynomial Accumulators. Tomescu [2, 3] formalizes representing a set S as the roots of $A(x) = \prod_{s \in S}(x - s)$ committed via KZG and analyzes membership and non-membership proofs in this setting. PGC differs from the Tomescu construction in four concrete ways that matter for graph workloads: *(i)* edge direction is encoded at the hash layer via domain-separated Blake3 with fixed-length ordered inputs (Eq. (1), (2)), so $H(u, v) \neq H(v, u)$ is forced by collision resistance rather than by application-layer convention; *(ii)* existence and non-existence share a single tagged 81-byte wire format, with the existence bit forcing $v = 0$ in verification; *(iii)* snapshots are chained into a Merkle tree with three distinct Blake3 domain separators for leaves, internal nodes, and padding, bound to the ceremony via a 32-byte setup fingerprint; and *(iv)* a label-mode verifier (§5.3) removes the position map from the verifier's trust base.

Authenticated Graphs and Provenance. Concurrent work on cloud-based endpoint auditing (vCause [25]) uses graph accumulators and verifiable provenance structures to authenticate causality queries. PGC is complementary: it commits to the underlying directed edge set with constant-size membership and non-membership proofs and can serve as the base accumulator beneath a provenance authenticator.

Batching. Boneh *et al.* [6] presented RSA/bilinear batching, and Bootle–Groth [19] batch zero-knowledge arguments for low-degree polynomials. BDFG21 [5] introduced efficient multi-point KZG openings; PGC uses this for batch edge proofs. ZKGraph [11] verifies graph queries, while PGC supplies the committed graph state beneath those queries.

General-Purpose ZK.. Groth16 [12] (∼192 B, relation-specific setup), PLONK (∼450 B), STARK [13] (100–200 KB, post-quantum) all require a custom circuit and 4–10 pairings for verification. PGC achieves 81 bytes and one pairing by exploiting the accumulator's polynomial structure directly.

Alternatives. Verkle trees [14]: $O(\log N)$ proofs, KV-map domain. RSA accumulators [15, 16]: $O(1)$ proofs but trusted modulus, no BDFG21 batch. Vector commitments [17]: ordered, positional—wrong primitive for graph edges.

3 PRELIMINARIES

Notation. $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$: groups of prime order p with bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Write $[x]_1 = x \cdot G_1$, $[x]_2 = x \cdot G_2$. $\mathbb{F}_p$: scalar field. We write $H(\cdot)$ for the edge hash function (Blake3 with a fixed domain separator, reduced modulo p); its formal definition appears in Equation (1).

BLS12-381. We use BLS12-381 [21]: ≈128-bit classical security after accounting for the extended tower number field sieve attack on pairing-friendly curves [22]. $\mathbb{G}_1$ elements: 48 bytes compressed. $\mathbb{F}_p$ elements: 32 bytes.

Definition 3.1 (KZG Polynomial Commitment [1]). *A KZG commitment to $P(x) \in \mathbb{F}_p[x]$, $\deg P \leq d$ is $C = [P(\tau)]_1$. Evaluation proof for $P(z) = v$: $\pi = [Q(\tau)]_1$, $Q(x) = (P(x) - v)/(x - z)$. Verification: $e(C - [v]_1, G_2) = e(\pi, [\tau - z]_2)$.*

Definition 3.2 (q-SDH Assumption). *Given $\{[\tau^i]_1\}_{i=0}^{q}$ and $[\tau]_2$, no PPT adversary outputs $(c, [1/(\tau + c)]_1)$ for any $c \in \mathbb{F}_p$ with non-negligible probability. We say the assumption holds if the advantage $\epsilon_{q\text{-SDH}}(\lambda)$ is negligible in λ.*

4 PGC CONSTRUCTION

4.1 Edge Hash Function

$$H(u, v) = \textsc{Blake3}\big(\texttt{"PGC\_EDGE"} \,\|\, \mathrm{pos}(u)_{64} \,\|\, \mathrm{pos}(v)_{64}\big) \bmod p \quad (1)$$

The domain separator prevents cross-protocol attacks. $H(u, v) \neq H(v, u)$ with overwhelming probability, preserving edge direction.

Field encoding. The Blake3 digest is reduced modulo the BLS12-381 scalar field prime to obtain a point in $\mathbb{F}_p$. The resulting distribution is not claimed to be perfectly uniform; this work relies on collision resistance and domain separation of the hash inputs rather than on a tight uniformity bound for the reduced outputs.

4.2 Accumulator Polynomial

$$P(x) = \prod_{e \in E}(x - H(e)), \quad C = [P(\tau)]_1.$$

Degree equals $|E|$; roots are exactly $\{H(e) : e \in E\}$. Built via a Rayon-parallelized product tree: FFT multiplication ($O(n \log n)$) for factors with ≥ 64 coefficients, naïve $O(n^2)$ for smaller ones. Overall: $O(|E| \log^2 |E|)$.

4.3 Existence Proof

Claim. $e \in E \Leftrightarrow P(H(e)) = 0$.

Proof structure. An `EdgeProof` consists of:

- $b \in \{0, 1\}$: existence flag (1 byte)
- $v \in \mathbb{F}_p$: $P(H(e)) = v$ (32 bytes; 0 iff $e \in E$)
- $\pi \in \mathbb{G}_1$: KZG quotient commitment (48 bytes compressed)

Total cryptographic payload: 81 bytes, constant regardless of $|E|$. In the position-based mode this payload is transmitted alongside query context. In the stateless label mode, the verifier additionally needs the endpoint labels as external query context or transport metadata so it can recompute $H^{\text{label}}(u, v)$ without a position map. The canonical 81-byte wire format intentionally omits these labels. With hiding mode (§6.3), the cryptographic payload is 114 bytes.

4.4 Non-Existence Proof

If $e \notin E, P(H(e)) = v \neq 0$. The prover outputs a standard KZG proof for $P(H(e)) = v$; the verifier checks $v \neq 0$ and the pairing equation. In the dedicated `NonExistenceProof` API the semantic claim is implicit, so the cryptographic payload is just $v \in \mathbb{F}_p$ (32 bytes) plus $\pi \in \mathbb{G}_1$ (48 bytes), i.e. 80 bytes. If a transport wants a single tagged format for both positive and negative answers, adding a one-byte existence flag yields the same 81-byte tagged payload used for `EdgeProof`. Either way, non-existence remains constant-size—a key advantage over Merkle-based schemes.

4.5 Verification

(1) *Semantic.* `exists` $\Rightarrow v = 0$; else $v \neq 0$.

(2) *Endpoint binding.* Recompute $H(u, v)$ from claimed labels. Prevents relabeling proof of (a, b) as proof of (c, d).
(3) *KZG pairing.* $e\big(C - [v]_1,\, G_2\big) = e\big(\pi,\, [\tau - H(u, v)]_2\big)$

4.6 Cycle Support

Vertex positions are assigned *before* edge hashing; no recursive dependency exists. $A \to B \to C \to A$ is committed as $(x - H(A, B))(x - H(B, C))(x - H(C, A))$, identical in treatment to any acyclic graph.

5 SECURITY MODEL AND FORMAL ANALYSIS

Posture. PGC is an application of KZG [1] and BDFG21 [5] openings to the edge-set accumulator polynomial $P(x) = \prod_{e \in E}(x - H(e))$. The theorems below do not reprove KZG or BDFG21 soundness; instead they state explicit reductions from each PGC security property to the standard q-SDH assumption (Def. 3.2) and to Blake3 collision resistance on the domain-separated input formats of Eq. (1)–(2).

Setup and trapdoor. All claims below assume an honestly generated KZG structured reference string $\mathsf{SRS} = \{[\tau^i]_1\}_{i=0}^{d} \cup \{[\tau]_2\}$ with unknown trapdoor $\tau \in \mathbb{F}_p$. If the ceremony is compromised, both commitments and openings can be forged. Hiding mode (§6.3) further requires an auxiliary generator $H_1 \in \mathbb{G}_1$ sampled so that $\log_{G_1}(H_1)$ is unknown.

5.1 Adversary Experiments

We formalize membership, non-membership, batch, and temporal soundness as game-based experiments. In each experiment the adversary $\mathcal{A}$ is a PPT algorithm given the SRS and chooses the edge set before the challenge; this captures chosen-graph attacks.

Definition 5.1 (Edge-Membership Soundness Experiment). $\mathsf{Expt}^{\mathrm{mem}}_{\mathsf{PGC},\mathcal{A}}(\lambda)$:

(1) $\mathsf{SRS} \leftarrow \mathsf{KZGSetup}(1^\lambda, d)$.
(2) $(E, \mathrm{aux}) \leftarrow \mathcal{A}(\mathsf{SRS})$ *with* $|E| \le d$.
(3) $C \leftarrow \mathsf{Commit}(\mathsf{SRS}, E)$.
(4) $(u, v, \pi) \leftarrow \mathcal{A}(\mathsf{SRS}, C, E, \mathrm{aux})$.
(5) Return 1 *iff* $\mathsf{Verify}(\mathsf{SRS}, C, (u, v), b{=}1, \pi) = 1$ *and* $(u, v) \notin E$.

Definition 5.2 (Edge Non-Membership Soundness Experiment). $\mathsf{Expt}^{\neg\mathrm{mem}}_{\mathsf{PGC},\mathcal{A}}(\lambda)$ *is defined identically except the adversary wins iff* $\mathsf{Verify}^{\neg}(\mathsf{SRS}, C, (u, v), v{\neq}0, \pi) = 1$ *and* $(u, v) \in E$.

Definition 5.3 (Batch Soundness Experiment). $\mathsf{Expt}^{\mathrm{batch}}_{\mathsf{PGC},\mathcal{A}}(\lambda)$: *after* $C \leftarrow \mathsf{Commit}(\mathsf{SRS}, E)$, $\mathcal{A}$ *outputs a batch witness* π_B *with per-edge claims* $\{(u_i, v_i, b_i, v_i^\star)\}_{i=1}^{n}$. $\mathcal{A}$ *wins iff* $\mathsf{Verify}^{\mathrm{batch}}$ *accepts and there exists an index* i *for which the claim disagrees with* E *(either* $b_i{=}1$ *but* $(u_i, v_i) \notin E$*, or the claimed evaluation* $v_i^\star$ *differs from* $P(H(u_i, v_i))$*).*

Definition 5.4 (Temporal Binding Experiment). $\mathsf{Expt}^{\mathrm{time}}_{\mathsf{PGC},\mathcal{A}}(\lambda)$: $\mathcal{A}$ *chooses an epoch chain* $\langle E_0, \ldots, E_{k-1}\rangle$ *and fresh epoch identifiers* $\eta_0, \ldots, \eta_{k-1}$*; each* $C_j = \mathsf{Commit}(\mathsf{SRS}, E_j)$ *is computed honestly and folded into the root* $R = \mathsf{MerkleRoot}(\ell_0, \ldots, \ell_{k-1})$ *with leaves* $\ell_j = \mathsf{Blake3}(\mathtt{PGC\_EPOCH\_LEAF} \| t_j \| \eta_j \| C_j \| |E_j|)$. $\mathcal{A}$ *then outputs a temporal proof* $(u, v, j, \pi, v^\star, \mathrm{path}, \mathrm{fp})$ *and wins iff* $\mathsf{Verify}^{\mathrm{time}}$ *accepts against* R *and the claim "*$(u, v) \in E_j$*" is false.*

5.2 Theorems and Proof Sketches

Theorem 5.5 (Edge-Membership Soundness). *Under the q-SDH assumption (Def. 3.2) with* $q \ge d$ *and the collision resistance of Blake3 on the* PGC_EDGE *(resp.* PGC_LABEL*) input format, for every PPT* $\mathcal{A}$*:*

$$\Pr\Big[\mathsf{Expt}^{\mathrm{mem}}_{\mathsf{PGC},\mathcal{A}}(\lambda) = 1\Big] \le \epsilon_{q\text{-}SDH}(\lambda) + \epsilon_{\mathrm{CR}}(\lambda) + \epsilon_{\bmod p}(\lambda) = \mathrm{negl}(\lambda).$$

Proof sketch. Let $z = H(u, v)$. Acceptance forces $b = 1$, hence the verifier accepts π as a KZG opening of $P(z) = 0$. Two cases.

Case (i). $z \notin \{H(e) : e \in E\}$. Then $P(z) \neq 0$ (since the only roots of P are the hashes of edges in E), yet π is an accepting opening claiming $P(z) = 0$. By the KZG opening soundness reduction [1, Thm. 3.1], extracting an adversary that produces accepting openings for two distinct evaluations at the same point yields an $(z, [1/(\tau{+}z)]_1)$ pair, contradicting the q-SDH assumption at degree $q = d$. A standard reduction converts our single forged opening into such a pair by running the honest prover on the true evaluation $y = P(z)$ and combining the two witnesses.

Case (ii). $z = H(e^\star)$ for some $e^\star \in E$ with $e^\star \neq (u, v)$. The adversary has produced two distinct ordered pairs mapping to the same field element under H. Because H prepends the fixed domain separator PGC_EDGE and uses fixed-length 8-byte little-endian position encodings (or fixed 32-byte Blake3 label digests in label mode), the inputs are injectively encoded: the only way to produce the same pre-reduction Blake3 output is a collision under the domain prefix. Beyond a Blake3 collision, two distinct outputs $h_1 \neq h_2 \in \{0, 1\}^{512}$ may also collapse to the same field element when $h_1 \equiv h_2 \pmod p$; modeling Blake3 as a random oracle on the domain-separated input format, the probability that two adversarial outputs coincide modulo p without colliding is bounded by $\lceil 2^{512}/p \rceil / 2^{512} \le 2/p$, so $\epsilon_{\bmod p}(\lambda) \le 2/p \le 2^{-253}$ for the BLS12-381 scalar field ($p > 2^{254}$). Union-bounding over the two cases and the two collision sources yields the claim. □

Theorem 5.6 (Edge Non-Membership Soundness). *Under the assumptions of Theorem 5.5, for every PPT* $\mathcal{A}$*:* $\Pr[\mathsf{Expt}^{\neg\mathrm{mem}}_{\mathsf{PGC},\mathcal{A}}(\lambda) = 1] = \mathrm{negl}(\lambda)$.

Proof sketch. Symmetric to Theorem 5.5. The non-membership branch of Verify requires a valid opening of $P(z) = y$ for some $y \neq 0$. If $(u, v) \in E$ then $z = H(u, v)$ is a root, so the true evaluation is 0; an accepting opening for a non-zero evaluation at a root is again an opening forgery, reducing to q-SDH as in Case (i) above. Collisions reduce to Blake3 collision resistance as in Case (ii). □

Theorem 5.7 (Endpoint Binding). *Fix a commitment* $C = \mathsf{Commit}(\mathsf{SRS}, E)$ *and an edge proof* π *accepted for endpoints* (u, v)*. Under q-SDH and Blake3 collision resistance on the corresponding hash domain, no PPT adversary can present distinct endpoints* $(u', v') \neq (u, v)$ *for which the same* π *is accepted, except with probability* $\mathrm{negl}(\lambda)$.

Proof sketch. Verification computes $z = H(u, v)$ and $z' = H(u', v')$ and checks the KZG pairing at each. If both accept with the same π, then either $z = z'$—which is a collision under the PGC_EDGE / PGC_LABEL domain separator—or π is an accepting opening at two distinct field points, contradicting KZG opening binding and hence q-SDH. □

Theorem 5.8 (Batch Soundness). *Under q-SDH with $q \geq d$ and Blake3 collision resistance, for every PPT $\mathcal{A}$:* $\Pr[\mathsf{Expt}^{\text{batch}}_{\mathsf{PGC},\mathcal{A}}(\lambda) = 1] = \text{negl}(\lambda)$.

Proof sketch. A PGC batch witness is a BDFG21 multi-point opening at $\{z_i = H(u_i, v_i)\}_{i=1}^{n}$ with claimed evaluations $\{v_i^\star\}$. By the BDFG21 opening soundness theorem [5, Thm. 1], an accepting batch witness implies $P(z_i) = v_i^\star$ for every i, except with probability negligible in λ under q-SDH at degree d. Hence any claim with $b_i = 1$ forces $v_i^\star = 0$, i.e. z_i is a root of P. By Theorem 5.5 Case (ii) every such z_i corresponds to an edge in E (up to a Blake3 collision event of probability $\text{negl}(\lambda)$), so the false claim required for the adversary to win cannot simultaneously satisfy acceptance. □

Theorem 5.9 (Temporal Binding). *Under q-SDH and Blake3 collision resistance on the* PGC_EPOCH_LEAF, PGC_EPOCH_NODE, *and* PGC_EPOCH_PAD *domains, for every PPT $\mathcal{A}$:* $\Pr[\mathsf{Expt}^{\text{time}}_{\mathsf{PGC},\mathcal{A}}(\lambda) = 1] = \text{negl}(\lambda)$.

Proof sketch. An accepting temporal proof for epoch index j triggers three independent checks: *(a)* a KZG opening of P_j at $z = H(u, v)$, *(b)* a Merkle path proving that $\ell_j = \mathsf{Blake3}(\texttt{LEAF} \| t_j \| \eta_j \| C_j \| |E_j|)$ hashes to R, and *(c)* a bitwise setup-fingerprint equality check against the verifier's SRS. Soundness of the opening reduces to Theorem 5.5 applied to C_j. Soundness of the path reduces to Blake3 collision resistance on the three epoch domain prefixes: any accepting path that commits to a different $(t_j, \eta_j, C_j, |E_j|)$ tuple than the honestly inserted leaf requires a collision on the LEAF prefix, on the NODE prefix (internal nodes), or on the PAD prefix (power-of-two padding). Finally, the fingerprint check rules out cross-ceremony swaps: a successful swap would require a second preimage on a Blake3 hash of the full serialized SRS. Union bound across the three events yields the claim. □

Theorem 5.10 (Hiding under Pedersen Blinding). *Let $C = [P(\tau)]_{G_1} + [B(\tau)]_{H_1}$ be the dual-generator hiding commitment with blinding polynomial $B(x)$ of degree at least t drawn with uniformly random coefficients in $\mathbb{F}_p$, where $\log_{G_1}(H_1)$ is unknown. Then for any two edge sets E_0, E_1 with $|E_0| = |E_1| \leq d$ and any fixed set of at most t opening points, the joint distribution of $(C, \pi_1, \ldots, \pi_t)$ under E_0 is statistically indistinguishable from that under E_1.*

Proof sketch. This is the standard Pedersen-blinded KZG hiding argument [1, 24]. A commitment C together with t evaluations $P(z_1), \ldots, P(z_t)$ specifies P modulo a $(d - t)$-dimensional affine subspace; the $t + 1$ random coefficients of B absorb the remaining information when viewed through the dual-generator commitment, because $B(\tau)$ is uniformly distributed over $\mathbb{F}_p$ conditioned on its values at any t points. Hence the transcript is distributed independently of the underlying edge set. Beyond t openings, B is reconstructible via Lagrange interpolation and hiding degrades gracefully; correctness of openings is unaffected. □

On the random-oracle heuristic. Theorems 5.5–5.9 invoke Blake3 only in its standard collision-resistance sense; we do not model Blake3 as a random oracle. The reduction to Blake3 collision resistance within each domain relies on the injectivity of the fixed-length encodings per domain prefix, which is explicit in the hash-input formats shown in §4–§6.

Setup-fingerprint scope. The 32-byte setup fingerprint (Blake3 of the serialized SRS) is a *binding* helper, not a relaxation of the trusted-setup assumption. It lets a verifier detect cross-ceremony artifact swaps, but a compromised trapdoor still yields forgeries against Theorems 5.5–5.9.

5.3 Stateless Verifier

Standard verification requires the position map Π_V to recompute $H(u, v)$. We introduce:

$$H^{\text{label}}(u, v) = \textsc{Blake3}\big(\texttt{"PGC\_LABEL"} \,\|\, \textsc{Blake3}(\text{label}_u) \,\|\, \textsc{Blake3}(\text{label}_v)\big) \bmod p \quad (2)$$

From the commitment, the SRS, the 81-byte proof payload, and endpoint labels supplied as external query context (or equivalent transport metadata), the verifier recomputes $H^{\text{label}}(u, v)$ and checks the same pairing. The in-memory `EdgeProof` struct carries labels for convenience, but the canonical wire format intentionally counts only the cryptographic payload. Test 16/16 covers this stateless roundtrip.

6 EXTENSIONS

6.1 BDFG21 Batch Proofs

For n edges: compute $Z(x) = \prod(x - z_i)$, interpolant $R(x)$ with $R(z_i) = v_i$, quotient $Q(x) = (P(x) - R(x))/Z(x)$, output $\pi = [Q(\tau)]_1$ and $C_R = [R(\tau)]_1$. Constant witness: $\pi + C_R = 96$ bytes. Per-edge: 33 bytes (v_i: 32 B, b_i: 1 B).

Table 1: BDFG21 Batch Proof Bandwidth

n	BDFG21	Individual	Saving
5	261 B	405 B	35.6%
10	426 B	810 B	47.4%
50	1,746 B	4,050 B	56.9%
100	3,396 B	8,100 B	58.1%
500	16,596 B	40,500 B	59.0%

Asymptotic saving: $(1 - 33/81) \approx 59.3\%$. More importantly, BDFG21 verifies n claims with *one pairing* vs. n pairings for individual proofs—an $n\times$ reduction in pairing count for verifiers. The verifier still performs $O(n)$ interpolation and MSM work to assemble $R(x)$ and $Z(\tau)$; the gain is in transmitted witness size and pairing cost, not in making total verifier arithmetic independent of batch size.

6.2 Temporal Proofs

A temporal proof for edge e at chain position j contains a KZG opening against P_j, the epoch root R, the setup fingerprint for the

ceremony used to create that epoch, and a Merkle path from ℓ_j to R, where $\ell_j = \text{BLAKE3}(\texttt{PGC\_EPOCH\_LEAF} \| t_j \| \eta_j \| C_j \| |E|_j)$, internal nodes use $\text{BLAKE3}(\texttt{PGC\_EPOCH\_NODE} \| \ell_L \| \ell_R)$, and power-of-two padding uses a dedicated $\text{BLAKE3}(\texttt{PGC\_EPOCH\_PAD})$ leaf hash. Here η_j is a unique epoch identifier generated when the snapshot is created. The proof also carries the chain index j used by the Merkle path, so duplicate timestamps remain unambiguous without changing the cryptographic payload size.

Excluding duplicated query context such as (u, v, t) and convenience metadata carried by the Rust struct (including `epoch_index` and `epoch_id`), the cryptographic payload for k epochs is

$$144 + 32\lceil \log_2 k \rceil \text{ bytes.}$$

This counts $\pi \in \mathbb{G}_1$ (48 B), $v \in \mathbb{F}_p$ (32 B), the epoch root (32 B), the 32-byte setup fingerprint, and the Merkle path.

6.3 Hiding Commitments

Dual-generator mode: $C = [P(\tau)]_{G_1} + [B(\tau)]_{H_1}$ where $B(x)$ is a random blinding polynomial and H_1 is a second independent generator in the style of Pedersen commitments [24]. Proof size with hiding: 114 bytes. The implementation exposes a caller-chosen blinding degree $t = \deg B$. Theorem 5.10 shows that fewer than $t + 1$ openings reveal no information about the underlying edge set, beyond what is already implied by the public evaluation points.

Setup trust model. PGC inherits the universal KZG trusted-setup assumption: soundness requires an honestly generated SRS whose trapdoor τ is unknown to the adversary. If the ceremony is compromised, commitments and openings can be forged. Dual-generator hiding mode further requires H_1 to be sampled so that $\log_{G_1}(H_1)$ is unknown. The setup fingerprint carried in serialized setups, epoch snapshots, and temporal proofs is a Blake3 hash of the full serialized setup, binding artifacts to one exact ceremony output; it does not remove the trusted-setup requirement.

7 IMPLEMENTATION

PGC is implemented in Rust (≈3,500 lines, `src/pgc.rs`) using `arkworks` (`ark-bls12-381`, `ark-ec`, `ark-poly`, `ark-ff`) and `blake3`.

Three-tier proof cache.

(1) **HOT.** Precomputed proofs for recurring queries.
(2) **WARM.** Cached quotient polynomials; skip $O(n)$ division.
(3) **COLD.** Full computation.

All in-memory caches are bounded and invalidated on any `add_edge`. The HOT proof cache uses FIFO eviction by entry count, while the WARM quotient cache uses a byte budget with FIFO eviction so adversarial unique misses cannot grow memory without bound. The precise microsecond ranges are hardware-dependent and reported from the repeated-sample cache microbenchmarks in §8.

Commit API hardening. The public API exposes a checked `try_commit()` path that returns an error when the loaded SRS is too small for the current graph. The legacy `commit()` wrapper remains as a convenience panic-on-misconfiguration entrypoint.

Setup serialization. Magic header `APGCSETP`; Blake3 fingerprint over the full serialized setup binds artifacts to the exact ceremony output and detects setup swaps.

Epoch log (APGCELOG).. Fixed 20-byte header (magic, version u32, count u64) followed by length-prefixed `ark-serialize` EpochSnapshot payloads. Mutations stage at a sibling `.tmp`, sync, and atomically rename—ensuring crash consistency.

8 EVALUATION

Setup. Intel i7-9700K (8 cores, 3.6 GHz), Linux 6.6 under WSL2 on a Windows 11 host, `rustc 1.91.0`, `cargo --release`. We report on a consumer-class baseline to keep the numbers reproducible on widely available hardware; server-class runs on similar silicon are typically 5–15% faster for the commit and prove columns, while verification—being pairing-dominated—is substantially hardware-portable. We make no hardware-lower-bound claim: the invariants we do claim are *size-independence* of verification latency and *constancy* of the 81-byte proof payload, both of which are architectural rather than microarchitectural. *Methodology.* Each scale is measured over N outer repetitions to capture run-to-run variance (CPU thermal state, allocator state, OS scheduling, cache pollution). Every outer rep rebuilds the graph from scratch, commits, then measures *commit*, *prove*, *verify*, and the three-tier *cold*/*warm*/*hot* cache paths independently. Reported numbers are **medians** across the N reps; the artifact bundle additionally records full [min, max] ranges per metric. Outer reps are scale-adaptive ($N = 10$ for $|E| \le 5\,\text{K}$, $N = 5$ for $|E| \le 100\,\text{K}$, $N = 3$ for $|E| > 100\,\text{K}$) so the 1 M-edge row stays practical while smaller scales get tighter medians. Within each rep, sub-microsecond operations are inner-averaged: each *verify* entry is averaged over 20 iterations at every scale, and the *prove* column uses 20 inner iterations through 10 K edges, 10 through 100 K, 3 through 500 K, and 1 at 1 M. A universal SRS up to the maximum measured degree is generated before the timed per-scale loop begins, so the reported commit/prove/verify rows exclude setup-generation time. The 1 M run still requires that setup to be present. The full artifact bundle, including the variance panels, is captured under `verify/reports/pgc_refresh_20260425_205435/`. The repository also ships a four-scale default benchmark for development spot-checks; Table 2 and the canonical artifact bundle refer to the explicit extended-scale rerun. The SRS is generated locally (single-party ceremony; production may reuse EIP-4844 or Perpetual Powers of Tau).

8.1 Proof Generation and Verification

Key observations. (i) **Proof size: constant 81 bytes** from 100 to 1 000 000 edges. (ii) Verification stays size-independent and in the low-millisecond band at 3.9–4.9 ms (median across reps), dominated by the fixed-cost BLS12-381 pairing checks rather than graph size. (iii) Commit time grows roughly linearly, reaching 48.1 s (median) at 1 M edges. (iv) Cold-path proving follows the same trend, reaching 31.2 s (median) at 1 M edges. (v) *Quantitative size-independence of verify.* Across the ten scales the verify medians have mean 4.37 ms and (sample) standard deviation 0.36 ms, a coefficient of variation of 8.3% across four orders of magnitude in $|E|$, with full spread [3.88, 4.88] ms. A log–log linear fit of verify time

Table 2: Measurement at scale (medians across outer reps; per-row [min, max] envelopes are recorded in the artifact bundle; all cryptographic proof payloads: 81 bytes)

Edges	Commit	Prove	Verify
100	46.0 ms	8.34 ms	4.88 ms
500	76.9 ms	30.1 ms	4.31 ms
1,000	94.2 ms	45.3 ms	4.06 ms
2,000	168.8 ms	96.4 ms	4.77 ms
5,000	351.3 ms	236.7 ms	4.33 ms
10,000	595.5 ms	423.1 ms	4.86 ms
50,000	2.52 s	1.82 s	4.43 ms
100,000	5.76 s	3.39 s	3.88 ms
500,000	24.6 s	16.1 s	3.99 ms
1,000,000	**48.1 s**	**31.2 s**	**4.14 ms**

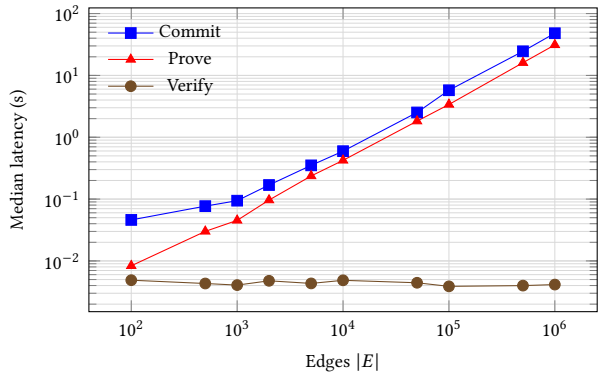

Figure 1: Median commit, prove, and verify latency vs. edge count $|E|$ on a log–log scale. Commit and prove track $|E|$ (log–log slopes 0.81 and 0.91, asymptoting to linear at the largest scales); verify is flat (slope −0.016, mean 4.37 ms, sample σ=0.36 ms, CoV 8.3%, spread [3.88, 4.88] ms). Source: verify/reports/pgc_refresh_20260425_205435/03_pgc_paper_benchmarks_extended.txt.

against $|E|$ has slope −0.016, statistically indistinguishable from zero and consistent with verify cost being dominated by a fixed pairing-batch budget rather than graph size. The same fit on commit and prove gives slopes 0.81 and 0.91, asymptoting to linear at the largest scales (where the per-call constant-cost component shrinks relative to the $\Theta(|E|)$ arithmetic). Figure 1 plots all three curves on log–log axes.

8.2 Cache Behavior

The implementation maintains HOT and WARM proof caches in addition to the cold path reported in Table 2. These caches are not new cryptographic objects—they serve the same 81-byte proof payload—but they remove repeated polynomial division and MSM work for recurring queries. In the median-aggregated April 25 rerun, the benchmark artifact reports amortized WARM hits in the 2.6–5.0 μs [min, max] envelope and amortized HOT hits at 0.1–0.2 μs across reps. Cold proof generation in the same rerun ranged from 7.2 ms (100 edges) to 34.7 s (1 M edges). The separate verification suite covers cache invalidation on mutation and hot/warm/cold roundtrips, but its one-shot stopwatch output is intentionally treated as regression evidence rather than as the published latency claim. The reproducibility bundle preserves the raw microbenchmark output for the current release baseline. Those amortized warm/hot figures come from repeated-sample cache probes with adaptive sample counts on the largest graphs (20/10/3/1 prove samples by size band; 100,000/20,000/5,000/2,000 warm-hot samples and 20/10/3/1 cold clones), so they should be read as steady-state cache-hit costs rather than end-to-end single-request latency.

8.3 Comparison with Commitment Schemes

Table 3: Membership proof footprint (PGC vs. alternatives)

Scheme	10K	100K	1M
Merkle (flat)	448 B	544 B	640 B
Sparse MT	8,192 B	8,192 B	8,192 B
Verkle ($k = 256$)	∼200 B	∼240 B	∼280 B
RSA accumulator	∼384 B	∼384 B	∼384 B
PGC	**81 B**	**81 B**	**81 B**

Baseline configurations. Merkle (flat): 32-byte Blake3 nodes, depth $\lceil \log_2 N \rceil$; size = $32\lceil \log_2 N \rceil$ B, computed analytically. Sparse MT: 32-byte Blake3 nodes, fixed depth 256 over the key-space hash domain; size = $32 \cdot 256$ B, constant, computed analytically. Verkle (k=256): vector commitments per [14] with width k=256; figures are order-of-magnitude estimates derived from the cited construction's asymptotic $O(\log_k N)$ depth. RSA accumulator: 3072-bit modulus (NIST 128-bit security level), single non-membership witness following [16]; figure is the cited witness size. PGC: BLS12-381 with 48-byte compressed $\mathbb{G}_1$ openings, 32-byte $\mathbb{F}_p$ evaluation, and a 1-byte existence tag (Theorem 5.5), measured directly by the artifact bundle. Hash and curve choices for all baselines are normalized to the BLS12-381 / Blake3 stack used by PGC; non-PGC numbers are not measured on identical hardware and should be read as scheme-comparative, not implementation-comparative.

Among the commitment baselines in Tables **??** and **??**, PGC is the only scheme in this comparison combining: (i) constant 81-byte proofs at any scale, (ii) native set semantics for directed edges with cycles, (iii) non-existence proofs at identical cost, and (iv) $O(1)$-pairing batch openings via BDFG21.

9 APPLICATIONS

We sketch three concrete applications that exercise distinct PGC capabilities. In each case the threat model is stated explicitly and the soundness guarantee is attributed to the relevant theorem from §5.

9.1 Verifiable Deletion Attestations (GDPR Article 17 Workflows)

Setting. A data controller C maintains a directed user-relationship graph $G = (V, E)$ committed as $C = \mathsf{Commit}(\mathsf{SRS}, E)$ and publishes the sequence of epoch commitments to an append-only Merkle-chained log (§6). A user u exercises the right to erasure over edges $E_u = \{e \in E : u \in e\}$.

Scope. The construction below produces a *verifiable deletion attestation*: a publicly verifiable certificate that E_u no longer appears in the controller's current committed graph C'. It is an audit-grade operational artifact supporting the controller's Article 17 workflow, not a cryptographic guarantee of erasure across third-party replicas, off-chain caches, or the structured reference string itself (which remains unchanged). Article 17 imposes broader obligations (downstream notification, residency of backups, de-indexing of derived datasets) that lie outside the cryptographic threat model and must be discharged by the controller's wider operational and legal processes.

Protocol.

(1) C computes $E' = E \setminus E_u$ and $C' = \mathsf{Commit}(\mathsf{SRS}, E')$.
(2) For each $e_i \in E_u$, C produces a `NonExistenceProof` π_i against C' certifying $e_i \notin E'$.
(3) C creates a snapshot $S = (t_{\text{now}}, \eta, C', |E'|)$ with fresh epoch id η, folds it into the epoch chain, and publishes the updated root R.
(4) The deletion attestation is $\text{DA} = (u, \{(e_i, \pi_i, v_i^\star)\}_i, S, \text{path}(S \to R), \sigma_C)$ where σ_C is a signature binding C to the attestation.

Verification. For every e_i: check the non-existence pairing against C' and the Merkle path from ℓ_S to R.

Threat model and guarantee. We assume an honestly generated SRS and a public, append-only epoch log. A malicious controller attempting to issue a fraudulent attestation while retaining e_i in the committed graph would have to produce an accepting non-existence proof for an edge that is still a root of P'; Theorem 5.6 rules this out except with probability $\text{negl}(\lambda)$. Theorem 5.9 additionally binds the attestation to the precise epoch (and thus the wall-clock time) at which C claims deletion occurred. We do not prevent a controller from refusing to publish C' altogether, nor from retaining copies of E_u outside the committed graph; those are liveness and operational properties outside the cryptographic threat model and must be enforced by auditing or regulatory process.

Bandwidth. An attestation for $|E_u|$ deleted edges has cryptographic payload $|E_u| \cdot 80\ \text{B} + 144 + 32\lceil \log_2 k \rceil\ \text{B}$ (with k the current chain length), independent of $|E'|$.

9.2 UBO Path Audits over Ownership Cycles

Setting. A corporate registry maintains an ownership graph whose edges encode direct shareholding relations. UBO (Ultimate Beneficial Owner) analysis frequently traverses cycles.

Protocol. Given a queried path $\mathcal{P} = (e_1, \ldots, e_n)$, the registry returns a BDFG21 batch proof $\pi_B = \mathsf{PGC.ProveBatch}(\mathcal{P})$ of constant 96-byte witness plus 33 B per edge.

Threat model and guarantee. Adversary: a registry administrator who wishes to forge inclusion of a spurious edge, or falsely assert non-inclusion of a real one, in $\mathcal{P}$. Theorem 5.8 forces any accepting batch to agree with the committed graph on every indexed edge (with probability $1 - \text{negl}(\lambda)$). Cyclic paths present no additional difficulty since PGC hashes each ordered pair independently (§4, Cycle Support).

9.3 Verifiable Grounding for Retrieval-Augmented Generation

Setting. An LLM-backed system serves responses citing facts drawn from a curated knowledge graph committed as C. Each cited fact (u, r, v) is represented as an edge in the directed graph.

Protocol. Each response carries one `EdgeProof` (81 B) per citation against the latest published C (or a batched BDFG21 witness for multi-fact responses).

Threat model and guarantee. The adversary is a compromised or hallucinating model that fabricates a citation not present in the knowledge graph. Verification in one pairing per edge (or one pairing per batch) catches any fabricated edge by Theorem 5.5, turning retrieval-attestation into a cryptographic check rather than a trust assumption on the generation pipeline.

10 LIMITATIONS AND FUTURE WORK

Trusted Setup. KZG requires a structured reference string. Transparent alternatives (FRI [18], Brakedown [20]) eliminate the ceremony at 10–100× larger proofs.

Post-Quantum Security. BLS12-381 is broken by Shor's algorithm. A post-quantum port would swap KZG for a lattice-based commitment [23] or FRI.

Commit-Time Scaling. Commit is $O(|E| \log^2 |E|)$ (product-tree construction), reaching 48.1 s (median) at $|E| = 10^6$ on our reference hardware (Table 2). Applications with high edge churn should amortize commit cost through batched rebuilds at epoch boundaries; truly incremental commit updates are future work.

Batch Verifier Arithmetic. BDFG21 reduces the pairing count to 1 for any batch size, but the verifier still performs $O(n)$ interpolation and MSM work to assemble $R(\tau)$ and $Z(\tau)$ for n batched edges. The proof-size and pairing-count invariants are thus genuinely constant; total verifier arithmetic is not.

Quotient Computation. Cold-path proof generation requires $O(|E|)$ synthetic division. Caches (§7) mitigate this for repeated queries.

Verifier Statefulness. [Solved in this work.] PGC-Label (§5.3) eliminates position-map dependency via content-addressed hashing.

Dynamic Updates. Edge mutations require rebuilding the accumulator. Incremental strategies remain future work.

11 CONCLUSION

We presented Polynomial Graph Commitments, a bilinear-accumulator commitment for directed graph edges. Domain-separated hashing, endpoint binding, single-pairing batches, temporal snapshots, and an 81-byte payload package, in a single directed-graph-edge commitment, a combination that to our knowledge no prior construction assembles together. Each capability individually has precedent: membership, non-membership, and batch openings from bilinear polynomial accumulators [3]; directed-graph size-independent proof systems from Yoshioka, Nakanishi, and Kitasuka [10]; and dynamic universal accumulation from Vitto and Biryukov [4]. PGC's

contribution is their graph-native commitment instantiation with endpoint binding, stateless label verification, temporal snapshot binding, and reproducible Rust evidence.

Empirical benchmarks confirm size-independent, low-millisecond verification latency and constant proof size from 100 to 1 000 000 edges. PGC fills the gap between Merkle trees ($O(\log N)$ proofs, no efficient non-existence) and RSA accumulators (no polynomial structure, no graph semantics), and is positioned as a viable construction for compliance, identity, and knowledge-graph applications.

REFERENCES

[1] A. Kate, G. M. Zaverucha, I. Goldberg, *Constant-Size Commitments to Polynomials and Their Applications*, ASIACRYPT, 2010.
[2] A. Tomescu, *Bilinear Accumulators for Cryptocurrency Enthusiasts*, Decentralized Thoughts, 2020.
[3] A. Tomescu, *Efficient Membership and Non-Membership Proofs from Polynomial Commitments*, IACR ePrint 2021/1045.
[4] G. Vitto, A. Biryukov, *Dynamic Universal Accumulator with Batch Update over Bilinear Groups*, Cryptographers' Track at the RSA Conference (CT-RSA), 2022; IACR ePrint 2020/777.
[5] D. Boneh, J. Drake, B. Fisch, A. Gabizon, *Efficient polynomial commitment schemes for multiple points and polynomials*, IACR ePrint 2020/081.
[6] D. Boneh, B. Bünz, B. Fisch, *Batching Techniques for Accumulators*, CRYPTO, 2019.
[7] T. Groß, *Certification and Efficient Proofs of Committed Topology Graphs*, IACR ePrint 2014/255; ACM CCSW, 2014.
[8] T. Groß, *Signatures and Efficient Proofs on Committed Graphs*, FC, 2015.
[9] S.-Y. Tan, I. Sfyrakis, T. Groß, *A Relational Credential System from q-SDH-based Graph Signatures*, IACR ePrint 2023/1181.
[10] T. Yoshioka, T. Nakanishi, T. Kitasuka, *Zero-Knowledge Proofs of Connectivity for Labeled Directed Graphs Using Bilinear-Map Accumulator*, IEICE Transactions on Fundamentals, Vol. E108-A, No. 11, 2025. DOI: 10.1587/transfun.2024EAP1166.
[11] H. Wu et al., *Zero-Knowledge Verifiable Graph Query Evaluation*, arXiv:2507.00427, 2025.
[12] J. Groth, *On the Size of Pairing-Based Non-Interactive Arguments*, EUROCRYPT, LNCS 9666, 2016.
[13] E. Ben-Sasson, I. Bentov, Y. Horesh, M. Riabzev, *Scalable, Transparent, and Post-Quantum Secure Computational Integrity*, IACR ePrint 2018/046.
[14] J. Kuszmaul, *Verkle Trees*, 2019.
[15] J. Benaloh, M. de Mare, *One-way accumulators*, EUROCRYPT, 1993.
[16] J. Camenisch, A. Lysyanskaya, *Dynamic accumulators and application to efficient revocation*, CRYPTO, 2002.
[17] D. Catalano, D. Fiore, *Vector Commitments and Their Applications*, PKC, 2013.
[18] E. Ben-Sasson et al., *Fast Reed–Solomon Interactive Oracle Proofs of Proximity*, ICALP, 2018.
[19] J. Bootle, J. Groth, *Efficient Batch Zero-Knowledge Arguments for Low Degree Polynomials*, PKC, 2018.
[20] A. Golovnev et al., *Brakedown: Linear-Time and Field-Agnostic SNARKs for R1CS*, CRYPTO, 2023.
[21] S. Bowe, *BLS12-381: New zk-SNARK Elliptic Curve Construction*, Electric Coin Co. Blog, 2017.
[22] T. Kim, R. Barbulescu, *Extended Tower Number Field Sieve*, CRYPTO, LNCS 9814, 2016.
[23] M. R. Albrecht et al., *Lattice-Based SNARKs*, CRYPTO, LNCS 13508, 2022.
[24] T. P. Pedersen, *Non-Interactive and Information-Theoretically Secure Verifiable Secret Sharing*, CRYPTO, LNCS 576, 1992.
[25] Q. Song et al., *vCause: Efficient and Verifiable Causality Analysis*, arXiv:2603.15216, 2026.

A REPRODUCIBILITY

```
# Regenerate the full paper evidence bundle:
./verify/run_pgc_paper_refresh.sh

# Or run the core commands directly:
cargo test --test pgc_paper_verification --release --
    --nocapture

cargo test --lib --release

PGC_PAPER_SCALES=100,500,1000,2000,5000,10000,\
50000,100000,500000,1000000 \
cargo test --test pgc_paper_benchmarks --release -- --
    nocapture
```

All 16 tests in `pgc_paper_verification.rs` pass in release mode, verifying: proof size constancy, accumulator semantics, Blake3 domain separation, cycle support, pairing soundness, non-existence proofs, BDFG21 batch constancy, temporal proofs, hiding commitments, cache invalidation, setup serialization, accumulator encoding preventing sum collisions, and PGC-Label stateless verifier roundtrip. The 20 library tests (`cargo test --lib --release`) and the extended-scale benchmark at 100–1,000,000 edges are regenerated by the same refresh script and archived alongside this paper's evidence bundle. This appendix intentionally refers only to properties with explicit tests; the formal theorems of §5 state the cryptographic guarantees those tests empirically sample.