ZINK: PRIVACY-PRESERVING AND VERIFIABLE INFLUENCE MAXIMIZATION IN SOCIAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Influence maximization (IM) is a fundamental problem in social networks, where the goal is to identify a set of seed nodes that can maximize the spread of influence. While many existing IM algorithms assume full access to the network structure or rely on online learning, these social graphs maintained by real-world platforms (e.g., TikTok) are often derived from sensitive user interactions and are proprietary. Despite having a strong incentive to help advertisers identify effective seed nodes, e.g., to promote their ad services, these platforms face significant constraints due to privacy concerns and intellectual property protections. As a result, the major users of IM algorithms often lack access to the underlying social network. To address this tension, we propose ZINK, a zero-knowledge proof (ZKP) framework for influence maximization. ZINK allows the platform to compute and prove a greedy IM result—the selected seed set and its influence spread—over a *proprietary yet cryptographically committed* social network, while the proof objects reveal no information about the private witness beyond the declared public statement. However, the IM problem is inherently complex, and naïve ZKP protocols are often impractical even for a moderately sized social network. We therefore introduce practical assumptions and optimizations that make ZKP for IM tractable. We implement ZINK and evaluate it on real-world datasets, showing that it scales to networks with thousands of nodes. Compared with general-purpose zero-knowledge virtual machines (zkVMs), ZINK achieves up to $42.72\times$ faster proof generation, with moderate verification cost and constant proof size after aggregation.

1 INTRODUCTION

Influence maximization (IM) selects a small seed set to maximize influence spread in a social network (Kempe et al., 2003; Chen et al., 2010), supporting applications such as viral marketing (Azaouzi et al., 2021). Advertisers and other influence seekers often lack access to the platform's private network structure and influence parameters. Delegating IM to the platform therefore creates a trust problem: it may manipulate the results, while disclosing the graph for independent verification would compromise privacy. This setting requires network privacy and verifiable IM computation.

Existing privacy-preserving IM methods address this setting only partly. IM methods based on differential privacy (DP) protect individual information by perturbing the computation, potentially reducing the influence spread achieved by the selected seeds (Rahimian et al., 2023; Hou et al., 2025; Unnithan et al., 2025); their privacy objective also differs from keeping the platform's entire graph hidden. Secure multi-party computation (MPC) approaches enable joint computation over private network data (Tassa & Bonchi, 2014), but do not by themselves provide influence seekers with a reusable public proof of faithful greedy execution over a fixed network. Zero-knowledge proofs (ZKPs) allow the platform to commit to a network snapshot and prove the returned seed set and spread, while the proof contents reveal nothing about the private witness beyond the declared public statement (Figure 1). However, greedy IM repeatedly simulates diffusion across many candidates, making proof generation expensive. General-purpose zero-knowledge virtual machines (zkVMs) can prove these computations (RISC Zero, 2025; Succinct Labs, 2025), but their instruction-level emulation incurs substantial time and memory overhead.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

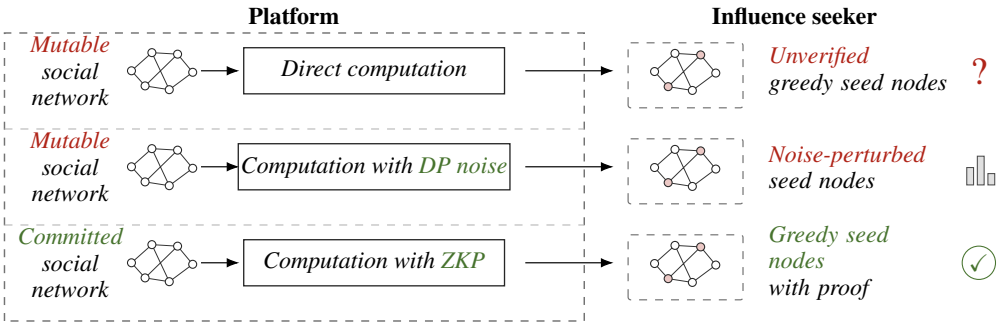


Figure 1: Comparison of approaches to IM on a private network. *Green text indicates desired properties; red text indicates limitations.*

To address these costs, we propose ZINK, a ZKP framework for deterministic greedy IM under the Linear Threshold (LT) diffusion model on a committed directed acyclic graph (DAG). Our design exploits the structure of both the network and the greedy computation to avoid unnecessary proof work. Network sparsity limits the dependencies that each diffusion step must check, while already activated nodes cannot increase spread when added as seeds, allowing redundant simulations to be replaced with cheaper checks. Within each greedy round, candidate evaluations are independent, so we divide them into smaller proving tasks that can run in parallel and combine their proofs while enforcing consistency with the same network and execution state. The resulting succinct proof lets downstream graph-based ML pipelines verify seed selection over private network data, providing an auditable optimization component. Experiments on synthetic and real networks show that ZINK scales to 5,000 nodes while matching plaintext greedy results, with proof generation before aggregation up to $42.72\times$ faster than the evaluated zkVMs. The source code of ZINK is available at https://anonymous.4open.science/r/zk-im-0030.

We summarize our contributions as follows:

- We propose ZINK, a zero-knowledge framework for trustworthy graph analytics that certifies greedy IM over a *cryptographically committed* social network, while revealing no witness information through the proof objects beyond the declared public statement.
- We introduce a set of algorithmic optimizations that make ZKP for IM tractable and efficient, supporting verifiable graph analytics within ML infrastructure.
- We implement ZINK and evaluate it on real-world datasets, showing that it scales to networks with thousands of nodes while preserving accuracy and strong privacy guarantees.

2 RELATED WORK

Influence maximization and private IM. Influence maximization was an important graph-based problem in Kempe et al. (2003), who established its hardness under the Independent Cascade and Linear Threshold (LT) models and the approximation guarantee of greedy selection. Subsequent systems accelerate influence estimation and seed selection through lazy evaluation, reverse-reachable sets, sketches, and learned models (Leskovec et al., 2007b; Tang et al., 2014; 2015; Cohen et al., 2014; Ling et al., 2023; Li et al., 2023b). These methods generally assume access to the graph. Privacy-preserving alternatives use differential privacy (Rahimian et al., 2023; Hou et al., 2025; Unnithan et al., 2025) or secure multiparty computation (Tassa & Bonchi, 2014). Differential privacy protects individuals by perturbing the computation and can reduce seed quality, while MPC targets jointly held inputs and does not by itself give a reusable public proof that a platform followed greedy IM on a fixed private graph. ZINK instead treats the platform as a prover and binds every query to a committed network snapshot.

Zero-Knowledge Proofs. Zero-knowledge proofs (ZKPs) are cryptographic protocols that allow a prover to convince a verifier that a statement is true without revealing any additional information beyond the validity of the statement (Goldwasser et al., 1989). ZKPs have been widely studied and applied in various domains, including blockchain (Sasson et al., 2014) and secure computa-

tion (Ozdemir & Boneh, 2022). General-purpose ZKP systems, such as zk-SNARKs (Groth, 2016), zk-STARKs (Ben-Sasson et al., 2018), and Bulletproofs (Bünz et al., 2018), allow the construction of ZKPs for arbitrary computations by compiling high-level programs into low-level circuits. Recent works have also explored optimizing ZKP systems for specific applications, such as machine learning (Chen et al., 2024), query processing (Li et al., 2023a) and data analysis (Wang et al., 2025). However, to the best of our knowledge, there has been no prior work on applying ZKPs to the influence maximization problem. In this work, we design a specialized ZKP framework, ZINK, tailored for influence maximization, incorporating various optimizations to handle the complexity of the problem efficiently.

3 PROBLEM SETUP AND GUARANTEES

3.1 INFLUENCE MAXIMIZATION

Influence maximization (IM) selects at most k initially active *seed nodes* to maximize the eventual spread of influence in a social network, where k is the seed budget. Let N be the number of nodes and $[N] = \{0, \dots, N-1\}$ their identifiers.

A diffusion model determines how influence propagates from the seeds. Under the Linear Threshold (LT) model, node i has an activation threshold θ_i and a list L_i of incoming edges; an entry $(j, \omega_{ji}) \in L_i$ records parent j and its nonnegative influence weight. Seeds start active, and an inactive node activates when the total weight from its active parents reaches its threshold. Active nodes remain active, and diffusion continues until no further activations occur. For fixed weights and thresholds (L, θ) and seeds $S \subseteq [N]$, define

$$\mathcal{A}(S; L, \theta) \subseteq [N], \qquad \sigma(S; L, \theta) = |\mathcal{A}(S; L, \theta)| \tag{1}$$

as the final active set, or LT *closure*, and its influence *spread*. Its indicator vector is $a(S; L, \theta) \in \{0,1\}^N$, with $a_i = 1$ exactly when $i \in \mathcal{A}(S; L, \theta)$.

Exact IM is NP-hard (Kempe et al., 2003). The standard greedy algorithm adds the node with the largest marginal gain at each round, resolving ties with a deterministic rule τ. Starting from $S^{(0)} = \varnothing$, round $t \in \{1, \dots, k\}$ chooses

$$v_t = \underset{v \in [N] \setminus S^{(t-1)}}{\arg\max}{}^{\tau} \sigma(S^{(t-1)} \cup \{v\}; L, \theta), \qquad S^{(t)} = S^{(t-1)} \cup \{v_t\}. \tag{2}$$

Maximizing this spread is equivalent to maximizing marginal gain because the current spread is common to every candidate. Write $\mathcal{A}^{(t)} = \mathcal{A}(S^{(t)}; L, \theta)$ and $a^{(t)} = a(S^{(t)}; L, \theta)$. The classical greedy approximation guarantee applies to expected LT spread under independent uniformly random thresholds; it does not automatically extend to a fixed-threshold instance (Appendix A.2).

3.2 NON-INTERACTIVE ZERO-KNOWLEDGE PROOFS

A zero-knowledge proof (ZKP) establishes that a computation is correct without revealing its private input. ZINK uses a non-interactive zero-knowledge argument (NIZK): for an efficiently decidable relation $\mathcal{R}$, the public statement s is valid when some private witness W satisfies $\mathcal{R}(s, W) = 1$.

Proof system. Before any network or query exists, a one-time setup takes the unary security parameter 1^λ and publishes

$$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda). \tag{3}$$

The parameter λ sets the cryptographic security level. The public parameters pp are shared by both parties, support the ZINK circuit family, and are independent of the network, query, and witness.

After setup, for each claimed public statement s, the prover uses its witness W to generate a proof, which the verifier checks without learning W:

$$\pi \leftarrow \mathsf{Prove}(\mathsf{pp}, s, W), \qquad b \leftarrow \mathsf{Verify}(\mathsf{pp}, s, \pi) \in \{0,1\}. \tag{4}$$

Acceptance means $b = 1$, and $\mathsf{negl}(\lambda)$ denotes a negligible probability. Public circuit inputs encode s, private inputs encode W, and the constraints hold exactly when $\mathcal{R}(s, W) = 1$. A *recursive* circuit verifies child proofs inside another circuit, reducing many local proofs to one root proof.

ZINK instantiates this interface with Barretenberg's UltraHonk backend for Noir (Aztec Protocol, 2025b), whose PLONKish arithmetization expresses $\mathcal{R}$ as polynomial constraints over the finite field $\mathbb{F}_p$, where p is a large prime. Section 3.3 states the corresponding security assumptions.

Standard proof-system properties. The following are defining properties of the NIZK: *Soundness* means that an accepted proof implies a witness W satisfying $\mathcal{R}(s, W) = 1$, except with probability $\mathsf{negl}(\lambda)$. *Completeness* means that an honest proof of any valid (s, W) is accepted except with negligible probability. *Zero-knowledge* means that the proof object can be simulated from the public statement and thus reveals nothing about W beyond that statement. Because ZINK recursively composes local proofs, its root proof must preserve all three properties; Appendix C establishes them for the composed relation.

3.3 THREAT MODEL AND SECURITY ASSUMPTIONS

The platform owns the private network (L, θ) and acts as a computationally bounded, potentially malicious prover; the influence seeker is an honest verifier. ZINK certifies faithful execution of deterministic greedy LT influence maximization and the resulting spread over one committed network snapshot. This guarantee concerns computation over that snapshot; external graph authenticity and global IM optimality are outside its scope.

Our instantiation based on PLONKish arithmetization (Gabizon et al., 2019; Chen et al., 2023) assumes a securely generated universal structured reference string (SRS) (Aztec Protocol, 2025a; 2020), collision-resistant hashes, binding and hiding commitments, and knowledge soundness, completeness, and zero-knowledge of the underlying and recursively composed proof systems. We write $\phi = \mathsf{commit}(m)$, suppressing private commitment randomness. Fiat–Shamir challenges use domain-separated transcript hashes modeled as random-oracle outputs (Fiat & Shamir, 1986).

A public statement comprises the query identifier, circuit configuration, network commitment, selected seeds, and final spread. The network, thresholds, permutation, and intermediate execution data form the private witness. All proofs must agree on the committed instance and execution state. Range constraints preserve bounded integer semantics within finite-field arithmetic. The complete statement, witness, and constraints, together with the formal security, permutation, and pruning-leakage analyses, appear in Appendices C and E.

4 ZINK

Proving greedy IM directly is expensive because each round evaluates many candidates, and each evaluation simulates diffusion over the private network. ZINK addresses this cost through three complementary ideas: make each diffusion evaluation cheaper, bind separate evaluations to the same hidden network and state, and replace redundant evaluations with authenticated activation checks. Sections 4.1–4.3 develop these design insights before Section 4.4 introduces the circuits and puts them together into the complete protocol.

4.1 EFFICIENT DIFFUSION OVER SPARSE NETWORKS

Compact network representation. Real social networks are sparse and have highly uneven degree distributions (Backstrom et al., 2006), but a dense $N \times N$ adjacency matrix charges the circuit for every absent edge. The LT activation test only needs the incoming neighbors of each node. ZINK therefore stores node i's incoming edges in the fixed-length list L_i, zero padded to the predetermined public in-degree cap $d_{\max}$. This reduces the encoding from $O(N^2)$ to $O(Nd_{\max})$ while keeping its shape fixed by public parameters.

Field-native influence accumulation. The activation test sums the weights of active parents and compares the result with θ_i. Representing these bounded integer weights and thresholds directly in $\mathbb{F}_p$ lets the circuit use its native arithmetic for the accumulation. This avoids the extra bit manipulation and overflow checks of the evaluated fixed-point encoding. Range checks and the no-wrap condition in Appendix C ensure that the field sum equals the intended integer sum and that the threshold comparison preserves its integer meaning.

Topological preprocessing. ZINK proves a fixed LT instance on a directed acyclic graph (DAG). Before committing the network and answering queries, the prover computes a topological ordering of the DAG off-circuit and reindexes its incoming-edge lists and thresholds accordingly. The proof checks that every real edge (j, i) satisfies $j < i$, so a claimed ordering cannot be accepted merely on the prover's assertion. Since every parent's activation state is final before its child is processed, each candidate evaluation needs only one forward pass, reducing diffusion work from $O(N^2 d_{\max})$ to $O(N d_{\max})$. This preprocessing is independent of the seed set and is reused across queries; Appendix B.2 gives the basic diffusion protocol and the full derivation of the topological optimization.

4.2 CRYPTOGRAPHIC COMMITMENT SCHEME

The malicious platform must not change its private network between candidate proofs or rounds. Zero knowledge hides each witness but does not force separate proofs to share one; public commitments provide this cross-proof binding. Here commit denotes the hiding and binding commitment primitive defined in Section 3. The Merkle-tree interface introduced below is a separate vector-commitment mechanism.

Network commitment. Before answering queries, the platform publishes $\phi_{\mathsf{net}} = \mathsf{commit}(\langle L, \theta \rangle)$ and proves that the committed representation satisfies the padding, index and value ranges, topological order, and no-wrap bound. Subsequent proofs carry this same digest, and each full diffusion evaluation checks its network opening. Binding prevents the platform from switching (L, θ) between candidates or rounds, while hiding protects the private snapshot. Thus, the network need only be fixed once, even though its candidate evaluations are proved separately.

Optimized node-set commitment. A node set $X \subseteq [N]$ is represented by its indicator vector $x \in \{0, 1\}^N$. Opening the entire vector to prove one membership fact would cost $O(N)$. ZINK instead uses a *Merkle tree* (MT), a binary tree whose leaves are the entries of x and whose internal nodes are collision-resistant hashes. Its root ρ binds the complete vector. The interface is:

- $\mathsf{MT.Commit}(x)$ builds the tree and returns its root ρ;
- $\mathsf{MT.Open}(x, i)$ returns leaf x_i and its sibling-hash authentication path μ_i;
- $\mathsf{MT.Verify}(\rho, i, x_i, \mu_i)$ recomputes the root and accepts exactly when that opening is consistent with ρ.

For the activation set after round t, ZINK sets $\rho_t = \mathsf{MT.Commit}(a^{(t)})$. Proving that a candidate v is already active then requires only a private authentication path for $a_v^{(t-1)} = 1$ under ρ_{t-1}, using $O(\log N)$ hashes. The root must be tied to the preceding greedy state: an opening under an arbitrary root would not justify skipping diffusion for the current seed set. Collision resistance prevents inconsistent openings, and consecutive rounds must share the same intermediate roots. The network commitment fixes the hidden instance, while the Merkle roots bind its evolving activation sets; together they establish consistency with the snapshot, not its external truth.

4.3 PRIVACY-PRESERVING PRUNING

Strategic Pruning via Activation Analysis. Even after making each diffusion evaluation cheaper, greedy IM repeatedly examines nodes whose marginal gain is already known. Two observations identify these redundant evaluations.

Seed Selection Monotonicity. If node v has been selected in a previous round, then $v \in S^{(t-1)}$ and adding it again leaves the seed set unchanged. Since greedy only adds seeds, its marginal gain remains zero in later rounds, and no new spread evaluation is needed for that node. It also remains ineligible for reselection under Equation 2.

Activation Exclusion. A node need not be a seed to have zero marginal gain: it may already belong to the final active set induced by the current seeds. The deterministic LT closure is monotone (adding seeds cannot deactivate nodes) and idempotent (diffusing again from the closure adds no nodes). Therefore,

$$v \in \mathcal{A}^{(t-1)} \implies \sigma(S^{(t-1)} \cup \{v\}; L, \theta) = \sigma(S^{(t-1)}; L, \theta). \tag{5}$$

The expensive spread evaluation can consequently be replaced by a membership check in the committed activation set. An active nonseed retains its zero-gain score for greedy selection, including

ties. The activation set used here is the closure of the already selected $S^{(t-1)}$, not the result of testing a new candidate in round t.

Privacy-Preserving Pruning via Random Permutation. Naively exposing which node receives a membership check instead of a full evaluation would reveal that node's activation status. To obscure this association, ZINK accesses candidates through a private permutation $\mathsf{perm} : [N] \to [N]$: slot $j \in [N]$ represents the actual node $v = \mathsf{perm}[j]$. A commitment $\phi_{\mathsf{perm}} = \mathsf{commit}(\mathsf{perm})$ binds all evaluations and selections to the same mapping. For a pruned slot, the proof privately resolves v and checks $\mathsf{MT.Verify}(\rho_{t-1}, v, 1, \mu_v) = 1$, where μ_v is the authentication path for its activation bit. This certifies zero marginal gain without revealing the slot-to-node mapping or the full activation vector. Every one of the N slots still requires a valid proof, so the platform cannot use pruning to omit a stronger candidate.

This combines the activation insight with the node-set commitment: membership checks cost $O(\log N)$ hashes each, but maintaining the permutation and activation roots adds overhead. Pruning is therefore most useful when many nodes are active; full evaluation can be cheaper in early rounds. The permutation hides the private mapping within proof contents, subject to the public statement described in Section 3.

4.4 PUTTING IT ALL TOGETHER

Table 1 summarizes the circuit realizations of the preceding design insights; +RP and +MT denote permutation lookup and activation-root construction.

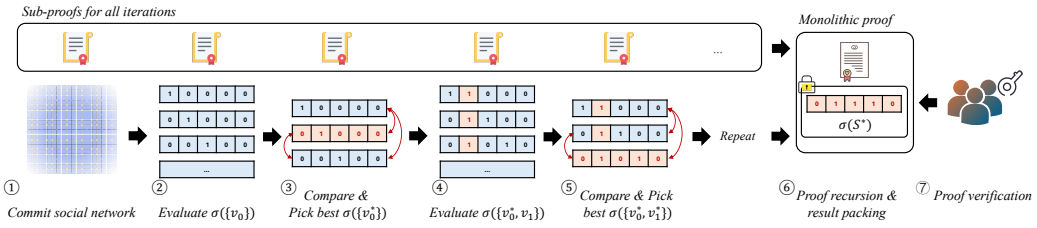


Figure 2: Workflow of ZINK. The final recursive proof binds the network commitment, candidate-evaluation proofs, and greedy-selection proofs to one state chain.

Execution. Figure 2 shows the composed execution. Commitment binds the validated network; PermGen binds a hash-derived Fisher–Yates shuffle (Fisher & Yates, 1938) using a fresh private salt. Each round certifies all candidate scores in parallel against one preceding state, using either unpermuted FullEval with Select or permutation-aware evaluation and pruning with Select+RP; indexing modes are never mixed. Selection excludes previous seeds, applies the public tie rule, and commits to the next state. Only the selected candidate's activation root supports subsequent pruning.

Composition and verification. During the execution, it generates two setup proofs, N candidate proofs and one selection proof per round, which give

$$M_{\mathsf{real}} = 2 + k(N + 1). \tag{6}$$

A width-w recursion tree verifies child proofs against their full public inputs, checking circuit tags, candidate coverage, common commitments, and consecutive-state equality; dummy proofs fill only surplus positions. The seeker verifies one root proof for s (Equation 17), with size independent of the leaf count. Detailed protocol, security arguments, recursion costs and implementation details appear in Appendices C, D and F.1.

5 EVALUATION

We evaluate ZINK in terms of end-to-end efficiency (RQ1), scalability (RQ2), accuracy preservation (RQ3), and the contribution of each optimization (RQ4).

Table 1: Circuit decomposition and measured constraint counts at $N = 1000$.

Circuit	Certified operation	Constraints
Commitment	Validate and bind the compact network and thresholds	$\sim$ 280K
PermGen	Bind a Fiat–Shamir Fisher–Yates permutation	$\sim$ 1034K
FullEval	Evaluate one unpermuted candidate	$\sim$ 2516K
FullEval+RP	Evaluate one permuted candidate	$\sim$ 2546K
FullEval+RP+MT	Evaluate and commit to its activation vector	$\sim$ 11229K
PruneEval	Prove zero marginal gain by activation membership	$\sim$ 36K
Select	Prove the unpermuted greedy argmax	$\sim$ 118K
Select+RP	Prove the permuted greedy argmax	$\sim$ 297K
ProofRecursion	Verify children and enforce statement chaining	$\sim$ 2252K

5.1 Experimental Setup

Datasets. We use synthetic and real networks. Synthetic directed acyclic graphs contain $N \in \{100, 200, 500, 1000, 2000, 5000\}$ nodes and follow the Barabási–Albert preferential-attachment model with $m = 5$ edges per arriving node (Barabási & Albert, 1999). The real datasets are Email-Eu-Core (1,005 nodes and 25,571 raw edges) (Leskovec et al., 2007a), Wiki-Vote (7,115 nodes and 103,689 raw edges) (Leskovec et al., 2010), and Hamsterster (2,426 nodes and 16,631 edges) (Kunegis, 2013). Because the current circuit requires a topological order, we break cycles before proving and evaluate plaintext baselines on the same resulting DAG. For Wiki-Vote, we use a 5,000-node subgraph and cap the maximum in-degree at 10. Unless stated otherwise, the seed budget is $k = 5$.

Baselines and configurations. Plaintext Greedy executes Algorithm 1 without proofs and serves as the exact accuracy reference. PrivIM (Hou et al., 2025) is the differentially private baseline; we use its default privacy budget $\epsilon = 3.0$ and report the spread achieved by its seeds relative to plaintext greedy. SP1 (Succinct Labs, 2025) and Risc0 (RISC Zero, 2025) are general-purpose zkVM baselines; each proves a Rust implementation of the same deterministic greedy LT computation using its standard SDK configuration. We report ZINK both as the collection of unaggregated subproofs and with recursive aggregation (ZINK +Rec.), which returns one root proof. The recursion width is chosen according to available memory: $w = 32$ up to $N = 1000$ and is reduced for larger networks, reaching $w = 10$ at $N = 5000$. A proving run is declared a timeout after 250 hours.

Environment and metrics. ZKP and plaintext experiments run on one partition of a high-performance-computing (HPC) cluster with 256 logical AMD EPYC 9754 cores and 768GB RAM. PrivIM runs on an NVIDIA RTX 5880 GPU. We measure end-to-end proof-generation time, verification time, transmitted proof size, peak memory, and influence spread. The accuracy comparisons in this section use the same preprocessed graph and deterministic tie rule; they do not compare against the globally optimal NP-hard seed set. Appendix F provides measurement details and separately evaluates preprocessing impact and expected-spread quality.

5.2 RQ1: End-to-End Performance

Table 2 gives the complete comparison. Without recursion, ZINK is $2.20\times$–$42.72\times$ faster to prove than SP1 and Risc0 at $N = 100$–500; with recursion it remains $1.61\times$–$37.54\times$ faster. Both zkVMs time out from $N = 1000$, whereas ZINK reaches $N = 5000$. Recursion adds 10.75% proving time at $N = 5000$ but reduces the transmitted proof from 169.48MB to 13.75KB and verification from 890.63s to 0.75s.

On the 100-node workload, the permutation-aware FullEval+RP circuit and the unpermuted FullEval circuit account for 79.5% and 11.5% of time, while commitment and permutation generation together contribute below 1%. Peak memory remains below 40GB at $N = 1000$, whereas SP1 already exceeds 40GB at $N = 100$. These results attribute the end-to-end advantage primarily to specialized evaluation circuits.

Table 2: End-to-end performance on synthetic networks with $k = 5$. TO denotes proof generation exceeding 250 hours; – denotes unavailable or inapplicable values. PrivIM and Plaintext report computation time only and do not generate or verify proofs. Accuracy is influence spread relative to plaintext greedy. Best time and proof-size values, including ties, are bold.

Method	Metric	Network size (N)					
		100	200	500	1,000	2,000	5,000
(a) Proof generation and computation time							
ZINK	Proof gen. (min)	**3.58**	**7.22**	**32.40**	**120.87**	**836.19**	**9,270.94**
ZINK +Rec.	Proof gen. (min)	4.89	9.03	36.87	156.34	1,046.91	10,267.45
SP1	Proof gen. (min)	7.89	41.51	526.59	TO	TO	TO
Risc0	Proof gen. (min)	13.79	136.51	1,384.01	TO	TO	TO
PrivIM	Runtime (s)	0.12	0.15	0.19	0.24	0.71	1.31
Plaintext	Runtime (s)	0.21	0.96	1.78	4.86	24.10	438.26
(b) Verification time, proof size, and accuracy							
ZINK	Proof ver. (s)	2.93	7.66	20.07	59.81	225.56	890.63
	Proof size (MB)	3.45	6.87	17.14	34.01	67.57	169.48
	Accuracy	100%	100%	100%	100%	100%	100%
ZINK +Rec.	Proof ver. (s)	0.69	0.69	0.71	**0.72**	**0.73**	**0.75**
	Proof size (KB)	13.75	13.75	13.75	**13.75**	**13.75**	**13.75**
	Accuracy	100%	100%	100%	100%	100%	100%
SP1	Proof ver. (s)	1.60	1.62	1.66	–	–	–
	Proof size (KB)	3.74	3.74	3.74	–	–	–
	Accuracy	100%	100%	100%	–	–	–
Risc0	Proof ver. (ms)	**6.83**	**6.84**	**6.86**	–	–	–
	Proof size (KB)	**0.65**	**0.65**	**0.65**	–	–	–
	Accuracy	100%	100%	100%	–	–	–
PrivIM	Proof ver.	–	–	–	–	–	–
	Proof size	–	–	–	–	–	–
	Accuracy	93.3%	78.7%	67.1%	54.0%	45.2%	33.8%
Plaintext	Proof ver.	–	–	–	–	–	–
	Proof size	–	–	–	–	–	–
	Accuracy	100%	100%	100%	100%	100%	100%

Answer to RQ1: ZINK provides practical proofs through 5,000 nodes; recursion trades a moderate proving overhead for constant 13.75KB proofs and sub-second verification.

5.3 RQ2: SCALABILITY

Figure 12 shows empirical $O(N^{2.50})$ growth with network size and approximately linear growth with k. Even at $k = 50$ on a 1,000-node graph, proving completes in under 913min. The modular candidate proofs also distribute naturally: for a greedy iteration on the synthetic 5,000-node workload, one, four, and eight 256-core machines require 1,854.19min, 477.54min, and 241.58min average time before final aggregation, corresponding to $3.88\times$ and $7.68\times$ multi-machine speedups.

The design also transfers beyond synthetic graphs. ZINK proves Email-Eu (1,005 nodes) in 102.19min, Hamsterster (2,426 nodes) in 1,629.36min, and the 5,000-node Wiki-Vote subgraph in 14,653.88min; their spreads cover 76.9%, 74.0%, and 35.9% of the nodes, respectively. Detailed node and edge counts appear in Table 6. These sizes cover community-level and targeted social subgraphs commonly used in IM studies (Leskovec et al., 2007a; Kunegis, 2013).

The different activation regimes make late-round pruning useful on Email-Eu and Hamsterster. The multi-machine experiment uses the synthetic 5,000-node workload, and reports per-iteration time.

Answer to RQ2: ZINK scales approximately linearly in k, reaches 5,000-node real and synthetic workloads, and obtains near-linear speedup from independent candidate proving across machines.

5.4 RQ3: ACCURACY PRESERVATION

Across all five datasets, ZINK selects the same ordered seeds and produces the same spread as plaintext greedy: for example, both reach 385 activated nodes on Synthetic-1000 and 1,796 on Hamsterster. PrivIM reaches only 208 and 1,363 nodes on those datasets, respectively. Its relative spread loss ranges from 20.7% on Email-Eu (613 versus 773 nodes) to 46.0% on Synthetic-1000, and reaches 35.5% on Wiki-Vote (1,156 versus 1,793 nodes), at $\epsilon = 3.0$. Table 7 reports all five comparisons. We can conclude that the proof transformation adds no algorithmic approximation. This comparison separates two privacy goals: PrivIM hides individual information statistically, whereas ZINK proves exact computation over a hidden committed network.

Matching the ordered trajectory is stronger than matching only the final spread: it checks candidate scoring, deterministic tie handling, and every state transition across all k rounds. The runtime columns should be interpreted accordingly. PrivIM performs GPU-based private prediction without producing a correctness proof, whereas ZINK and the zkVMs include cryptographic proving. The comparison therefore exposes their different accuracy and verifiability tradeoffs instead of treating them as identical services.

Answer to RQ3: ZINK preserves the complete greedy trajectory and final spread, while the evaluated DP baseline exhibits a measurable privacy-utility tradeoff.

5.5 RQ4: OPTIMIZATION EFFECTIVENESS

Table 8 identifies the one-pass topological evaluation as the dominant optimization: relative to a dense representation, it reduces proving time by $205\times$. The dense matrix and repeated diffusion designs also run out of memory at larger scales. Figure 15 gives the per-round pruning measurements. During the early rounds, checking whether pruning applies costs more than it saves, so ZINK uses FullEval. Adaptive pruning raises the total speedup to $258\times$. On Synthetic-1000, it becomes worthwhile in round 8 to round 9, and its per-round saving grows to 19.9% by round 50, reducing total proving time by 14.1%. Field-native arithmetic independently reduces one-iteration opcodes from roughly 343M to 130M and time from 7,232.40s to 1,450.72s, a 62.1% opcode reduction and $4.98\times$ speedup over the evaluated 32-bit fixed-point encoding.

Answer to RQ4: Compact storage is necessary, topological one-pass evaluation produces the largest gain, and pruning plus field-native arithmetic provide additional complementary savings.

6 CONCLUSION

We introduce ZINK, a zero-knowledge proof framework that enables privacy-preserving and verifiable influence maximization. Through a combination of compact network representation, field-native arithmetic, cryptographic commitments, and privacy-preserving pruning, ZINK allows a platform to compute greedy IM over a cryptographically committed social network and prove that the reported result follows from that fixed snapshot. Our evaluation demonstrates that ZINK scales to networks with thousands of nodes, outperforming general-purpose zkVMs by $2.20\times$–$42.72\times$ in proof generation while maintaining 100% accuracy relative to plaintext greedy algorithms. The resulting component supports trustworthy graph analytics by making delegated seed selection auditable, and can serve as an optimization layer in ML decision pipelines over proprietary relational data. Future work will support other diffusion models, optimize larger networks, and extend verifiable computation to other privacy-sensitive graph analyses.

ETHICS STATEMENT

ZINK is intended to reduce confidentiality and integrity risks when influence maximization is delegated to a platform that holds a private social graph. Our experiments use synthetic graphs and the graph structure of three publicly available benchmark datasets; we collect no new personal data, conduct no human-subjects study, and do not attempt to identify individuals. Nevertheless, influence maximization can support targeted advertising or information campaigns and can be used manipulatively. A valid ZINK proof certifies only that the declared greedy computation was executed over a committed snapshot; it does not establish that the campaign, chosen seeds, graph construction, or underlying data collection is fair, lawful, consensual, or socially beneficial. Moreover, zero-knowledge protects the private witness relative to the declared public statement, not the selected seed identifiers, final spread, or external timing, traffic, and scheduling metadata when those are released. Any deployment should therefore use authorized data, minimize public outputs, protect auxiliary metadata, and apply appropriate consent, governance, fairness, and legal safeguards. Large proofs also require substantial compute; Section 5 and Appendix F report runtime, hardware, and memory costs so that this resource burden can be assessed.

REPRODUCIBILITY STATEMENT

The anonymous source-code release is linked in Section 1. Sections 3 and 4 specify the problem instance, public and private inputs, security assumptions, data representation, circuit relations, and recursive composition. Section 5 reports the datasets and preprocessing summary, baseline configurations, parameters, hardware, timeout rule, and evaluation metrics. Appendices A–E provide the background algorithms, detailed circuit protocols, composed security proof, aggregation-overhead analysis, and formal permutation-randomness argument. Appendix F further records proving-key and proof-size accounting, original and processed graph sizes, the distributed-run configuration, preprocessing impact, detailed accuracy results, ablations, memory use, and per-round pruning measurements. Together, the source release and these specifications support independent reproduction of the protocol, proofs, and reported evaluation.

AI USE STATEMENT

This paper utilized AI-assisted tools during its preparation. Specifically, large language models were used to assist with editing and polishing of manuscript text across all sections for grammar, clarity, and presentation. We also use AI to generate part of the code for the implementation of the protocol. All AI-generated content was reviewed, verified, and revised by the authors. The authors take full responsibility for the correctness and originality of the submitted content.

REFERENCES

Mehdi Azaouzi, Wassim Mnasri, and Lotfi Ben Romdhane. New trends in influence maximization models. *Computer Science Review*, 40:100393, 2021.

Aztec Protocol. Ignition: Ceremony transcript verification, 2020. URL https://github.com/AztecProtocol/ignition-verification. Ceremony provenance, trust assumption, and transcript-verification procedure. Accessed: 2026-09-26.

Aztec Protocol. Barretenberg: BN254 SRS distribution, version 3.0.0-nightly.20251104, 2025a. URL https://github.com/AztecProtocol/aztec-packages/blob/v3.0.0-nightly.20251104/barretenberg/scripts/download_bb_crs.sh. Transcript format: https://github.com/AztecProtocol/aztec-packages/blob/v3.0.0-nightly.20251104/barretenberg/trusted_setup.md. Accessed: 2026-09-26.

Aztec Protocol. Barretenberg: UltraHonk backend, version 3.0.0-nightly.20251104, 2025b. URL https://github.com/AztecProtocol/aztec-packages/releases/tag/v3.0.0-nightly.20251104. Versioned implementation and release. Accessed: 2026-09-26.

Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 44–54, 2006.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286 (5439):509–512, 1999.

Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *Cryptology ePrint Archive*, 2018.

Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE symposium on security and privacy (SP)*, pp. 315–334. IEEE, 2018.

Bing-Jyue Chen, Suppakit Waiwitlikhit, Ion Stoica, and Daniel Kang. Zkml: An optimizing system for ml inference in zero-knowledge proofs. In *Proceedings of the Nineteenth European Conference on Computer Systems*, pp. 560–574, 2024.

Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. Hyperplonk: Plonk with linear-time prover and high-degree custom gates. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 499–530. Springer, 2023.

Wei Chen, Yifei Yuan, and Li Zhang. Scalable influence maximization in social networks under the linear threshold model. In *2010 IEEE international conference on data mining*, pp. 88–97. IEEE, 2010.

Edith Cohen, Daniel Delling, Thomas Pajor, and Renato F Werneck. Sketch-based influence maximization and computation: Scaling up with guarantees. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pp. 629–638, 2014.

Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pp. 186–194. Springer, 1986.

Ronald Aylmer Fisher and Frank Yates. Statistical tables for biological, agricultural and medical research. 1938.

Ariel Gabizon, Zachary J Williamson, and Oana Ciobotaru. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *Cryptology ePrint Archive*, 2019.

Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

Lorenzo Grassi, Dmitry Khovratovich, and Markus Schofnegger. Poseidon2: A faster version of the poseidon hash function. In *International Conference on Cryptology in Africa*, pp. 177–203. Springer, 2023.

Jens Groth. On the size of pairing-based non-interactive arguments. In *Annual international conference on the theory and applications of cryptographic techniques*, pp. 305–326. Springer, 2016.

Renxuan Hou, Qingqing Ye, Xun Ran, Sen Zhang, and Haibo Hu. Privim: Differentially private graph neural networks for influence maximization. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, pp. 3467–3479. IEEE Computer Society, 2025.

David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 137–146, 2003.

Jérôme Kunegis. Konect: the koblenz network collection. In *Proceedings of the 22nd international conference on world wide web*, pp. 1343–1350, 2013.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2–es, 2007a.

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 420–429, 2007b.

Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Signed networks in social media. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 1361–1370, 2010.

Xiling Li, Chenkai Weng, Yongxin Xu, Xiao Wang, and Jennie Rogers. Zksql: Verifiable and efficient query evaluation with zero-knowledge proofs. *Proceedings of the VLDB Endowment*, 16 (8), 2023a.

Yandi Li, Haobo Gao, Yunxuan Gao, Jianxiong Guo, and Weili Wu. A survey on influence maximization: From an ml-based combinatorial optimization. *ACM Transactions on Knowledge Discovery from Data*, 17(9):1–50, 2023b.

Chen Ling, Junji Jiang, Junxiang Wang, My T Thai, Renhao Xue, James Song, Meikang Qiu, and Liang Zhao. Deep graph representation learning and optimization for influence maximization. In *International conference on machine learning*, pp. 21350–21361. PMLR, 2023.

Alex Ozdemir and Dan Boneh. Experimenting with collaborative {zk-SNARKs}:{Zero-Knowledge} proofs for distributed secrets. In *31st USENIX Security Symposium (USENIX Security 22)*, pp. 4291–4308, 2022.

M Amin Rahimian, Fang-Yi Yu, and Carlos Hurtado. Differentially private network data collection for influence maximization. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pp. 2795–2797, 2023.

RISC Zero. Risc zero. https://github.com/risc0/risc0, 2025.

Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE symposium on security and privacy*, pp. 459–474. IEEE, 2014.

Succinct Labs. Sp1. https://github.com/succinctlabs/sp1, 2025.

Youze Tang, Xiaokui Xiao, and Yanchen Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 75–86, 2014.

Youze Tang, Yanchen Shi, and Xiaokui Xiao. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pp. 1539–1554, 2015.

Tamir Tassa and Francesco Bonchi. Privacy preserving estimation of social influence. In *17th International Conference on Extending Database Technology, EDBT 2014*, pp. 559–570. OpenProceedings. org, University of Konstanz, University Library, 2014.

Niranjana Unnithan, Balasubramaniam Natarajan, and George Amariucai. On random seeding for influence maximization in differentially private graphs: Balancing privacy and utility using percolation theory. *IEEE Transactions on Network Science and Engineering*, 2025.

Zhaoyu Wang, Pingchuan Ma, Zhantong Xue, Yanbo Dai, Zhenlan Ji, and Shuai Wang. Privacy-preserving and verifiable causal prescriptive analytics. *Proceedings of the ACM on Management of Data*, 3(6):1–27, 2025.

A Formal Background and Vanilla Baseline

A.1 Linear Threshold Diffusion

This appendix expands the LT semantics used by the main text into the exact recurrence enforced by the circuits. Let $[N] = \{0, \ldots, N-1\}$ and let (L, θ) be the compact representation from Section 3. For an activation vector $a \in \{0,1\}^N$ and node i, define the accumulated influence

$$I_i(a; L) = \sum_{(j,\omega_{ji}) \in L_i} \omega_{ji} a_j. \tag{7}$$

Starting from $a_i = 1$ exactly for $i \in S$, LT diffusion repeatedly sets $a_i \leftarrow 1$ whenever $I_i(a; L) \geq \theta_i$. The least fixed point is the closure $\mathcal{A}(S; L, \theta)$ in Equation 1. Because the committed graph is a DAG and L follows its topological order, one forward pass reaches this fixed point: every a_j used while evaluating i is already final. The closure is monotone and idempotent,

$$S \subseteq \mathcal{A}(S; L, \theta), \qquad \mathcal{A}(\mathcal{A}(S; L, \theta); L, \theta) = \mathcal{A}(S; L, \theta). \tag{8}$$

A.2 Greedy Influence Maximization

Algorithm 1 gives the deterministic computation certified by ZINK. The tie rule τ is part of the public configuration, so two honest executions select the same ordered seeds even when candidates have equal spread.

Algorithm 1: Greedy LT influence maximization

Input: (L, θ), seed budget k, tie rule τ
Output: $S^{(k)}$ and $\sigma(S^{(k)}; L, \theta)$

$S^{(0)} \leftarrow \varnothing$;
for $t \leftarrow 1$ **to** k **do**
  Evaluate $\sigma(S^{(t-1)} \cup \{v\}; L, \theta)$ for every $v \in [N] \setminus S^{(t-1)}$;
  $v_t \leftarrow \arg\max^{\tau}_{v \in [N] \setminus S^{(t-1)}} \sigma(S^{(t-1)} \cup \{v\}; L, \theta)$;
  $S^{(t)} \leftarrow S^{(t-1)} \cup \{v_t\}$;
end
return $S^{(k)}$, $\sigma(S^{(k)}; L, \theta)$;

Scope of the classical guarantee. Under the classical LT model, each node independently draws its threshold uniformly from $[0, 1]$, incoming weights are nonnegative and sum to at most one per node, and the objective is expected spread over these threshold draws (Kempe et al., 2003). This expected-spread objective is monotone and submodular, so greedy selection with exact marginal evaluations achieves the $(1 - 1/e)$ approximation guarantee; sufficiently accurate Monte Carlo estimates give the corresponding $(1 - 1/e - \epsilon)$ guarantee. These statements do not establish the same bound for an arbitrary fixed threshold assignment. ZINK certifies each greedy choice and the exact spread on one committed deterministic instance, without relying on a global approximation guarantee. Appendix F.3 compares the expected influence and computation time of fixed-threshold and Monte Carlo greedy empirically.

A.3 Revisiting Vanilla Greedy IM Algorithm

The greedy algorithm for influence maximization is a classic heuristic that incrementally builds a seed set by selecting, at each step, the node that provides the largest marginal gain in influence spread. Algorithm 2 expands the spread-evaluation step of Algorithm 1 into the vanilla dense-graph diffusion procedure under the LT model. Here ω is the dense weight matrix of the same network, and the public rule τ resolves ties. To assess the feasibility of a direct ZKP realization, we revisit the algorithm and analyze the computational complexity of influence spread computation and greedy selection.

The influence spread $\sigma(S, \omega, \theta)$ is computed by simulating the diffusion process on the graph ω, given a seed set S and activation thresholds θ. Although diffusion often terminates in fewer than

Algorithm 2: Vanilla greedy IM with dense diffusion evaluation

Input: ω: social network, represented as a weighted directed graph, θ: node thresholds, k: number of seed nodes to select, τ: public tie rule
Output: S: selected seed set, $\sigma(S)$: influence spread of S

$S \leftarrow \emptyset$;
while $|S| < k$ **do**
  $v^* \leftarrow \arg\max^{\tau}_{v \notin S} \sigma(S \cup \{v\}, \omega, \theta)$;
  $S \leftarrow S \cup \{v^*\}$;
end
return S, $\sigma(S, \omega, \theta)$
Function σ (S, ω, θ) **:**
  activated_nodes $\leftarrow S$;
  $N \leftarrow$ number of nodes in ω ;
  for $i \leftarrow \{1, \ldots, N\}$ **do**
    for $v \in \omega$ **do**
      influence $\leftarrow 0$;
      for $u \in \omega$ **do**
        if $u \in$ *activated_nodes* **then**
          influence $\leftarrow$ influence + ω_{uv} ;
      end
      if *influence* $\geq \theta_v$ **then**
        activated_nodes $\leftarrow$ *activated_nodes* $\cup\{v\}$;
    end
  end
  return |*activated_nodes*|

N rounds, a direct fixed-size circuit encoding reserves the worst-case execution path: it unrolls all N rounds regardless of early termination. Furthermore, while a conventional implementation can visit only active parents of a node, the dense circuit evaluates all N possible incoming contributions and masks them by activation bits to avoid witness-dependent branching. Processing all N nodes therefore costs $O(N^2)$ per round and $O(N^3)$ per spread evaluation. The greedy procedure performs k iterations, each requiring $O(N)$ candidate evaluations, for a total of $O(kN^4)$ operations, or $O(N^4)$ when k is constant. These bounds describe this vanilla encoding; they motivate changing the representation and evaluation order.

This computational burden is prohibitive for practical ZKP systems. For example, with $N = 10^3$ and $k = 5$, a naïve dense circuit would involve trillions of arithmetic operations, creating substantial proving-time and memory costs. Partitioning the greedy process into smaller circuits enables parallel proof generation, but does not by itself remove the $O(kN^4)$ total work. Appendix B.2 therefore develops the basic compact-list protocol and the assumptions and optimizations that reduce its cost: compact lists first replace dense scans, topological ordering removes repeated diffusion rounds on a DAG, and activation-based pruning avoids redundant candidate evaluations.

A.4 Non-Interactive Zero-Knowledge Arguments

For an efficiently decidable relation $\mathcal{R}$, a statement s is valid if there exists a witness W such that $\mathcal{R}(s, W) = 1$. A non-interactive proof system consists of setup, proving, and verification algorithms:

$$\begin{aligned} \mathsf{pp} &\leftarrow \mathsf{Setup}(1^\lambda), \\ \pi &\leftarrow \mathsf{Prove}(\mathsf{pp}, s, W), \\ b &\leftarrow \mathsf{Verify}(\mathsf{pp}, s, \pi). \end{aligned} \tag{9}$$

Reliability (completeness) accepts honest proofs of valid statements; knowledge soundness prevents acceptance of a false statement except with probability $\mathsf{negl}(\lambda)$ and provides witness extraction; zero-knowledge supplies a simulator for proof objects from their public statements. Under the PLONKish instantiation and assumptions stated in Section 3.3, the computation is expressed

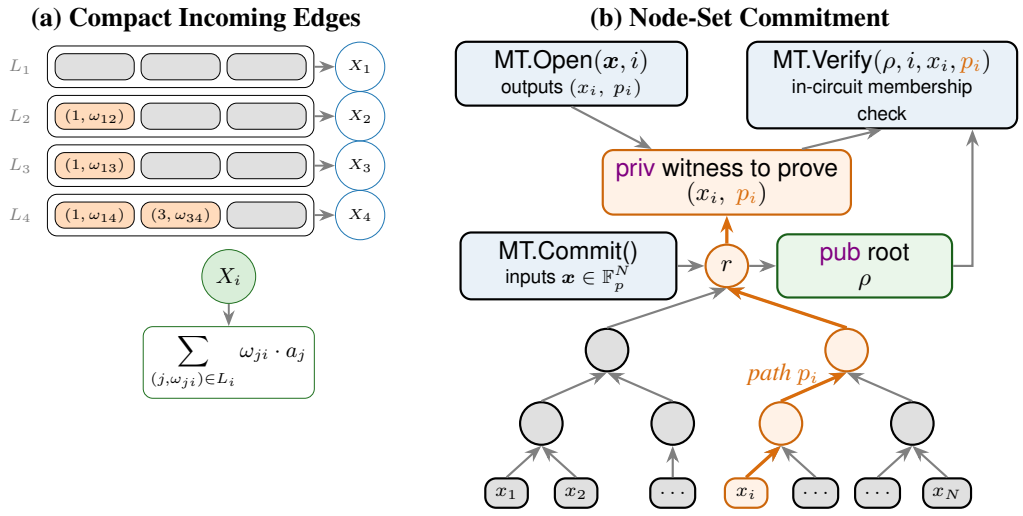


Figure 3: Compact network representation and node-set commitment. (a) Compact incoming-edge lists. (b) Node sets committed via a Merkle tree root ρ.

as polynomial constraints over $\mathbb{F}_p$. Appendix C instantiates s, W, and $\mathcal{R}$ for the composed ZINK protocol.

B DETAILED ZINK DESIGN

B.1 REPRESENTATION AND COMMITMENT INTERFACES

The network uses N fixed-length incoming-edge lists. A real entry of L_i stores (j, ω_{ji}) with $j < i$ in the committed topological order; unused entries are zero padded. The Commitment relation checks the list length, padding syntax, node-index ranges, topological condition, value ranges, and no-wrap bound before producing $\phi_{\mathsf{net}} = \mathsf{commit}(\langle L, \theta \rangle)$. This reduces storage from $O(N^2)$ to $O(N d_{\max})$ and aligns the witness with the access pattern of Equation 7.

Degree cap and preprocessing. The public $d_{\max}$ is a predetermined capping limit used to size the circuit, rather than the measured maximum degree of the source graph. Before commitment, incoming edges of any node exceeding this limit are capped; shorter lists are zero padded to the same length. Publishing this fixed limit therefore does not disclose the source graph's actual maximum in-degree or which nodes were capped. The public configuration still discloses the declared graph size N and the chosen capacity bound. Cycle breaking and degree capping change the instance being proved; Appendix F.2 measures their combined effect on influence spread.

Intermediate vectors use a binary Merkle tree with a field-native collision-resistant hash. Its interface is:

- $\mathsf{MT.Commit}(x)$ constructs the tree for $x \in \mathbb{F}_p^N$ and returns its root ρ;
- $\mathsf{MT.Open}(x, i)$ returns x_i and the $O(\log N)$ sibling path μ_i;
- $\mathsf{MT.Verify}(\rho, i, x_i, \mu_i)$ recomputes the root and accepts exactly when the opening is valid, up to hash collision probability.

Every commitment and local result is domain-separated by (qid, t, j), preventing a proof or opening from one query, round, or candidate slot from being replayed in another.

B.2 INFLUENCE SPREAD COMPUTATION AND OPTIMIZATION STRATEGIES

The influence spread $\sigma(S, \omega, \theta)$ counts the nodes activated by the cascade initiated by seed set S. Using the compact representation from Section 4, we write the same computation as $\sigma(S, L, \theta)$, where L stores incoming edges and θ stores activation thresholds. We first describe a basic protocol

that replaces dense scans with these lists, then develop topological evaluation and privacy-preserving pruning to reduce the cost further.

B.2.1 Basic Algorithm Design

Protocol 3: Basic Influence Spread Computation

Protocol σ (*pub* S, *priv* L, *priv* θ) **:**
- $\phi \leftarrow \mathsf{commit}(\langle L, \theta \rangle)$; `// Network commitment`
- $\mathcal{A} \leftarrow S$; `// Initialize activated node set`
- $N \leftarrow |L|$; `// Total number of nodes`
- **for** *round* $\leftarrow 1$ **to** N **do**
  - **for** $i \leftarrow 0$ **to** $N-1$ **do**
    - $\text{influence}_i \leftarrow 0$;
    - **for** $(j, \omega_{ji}) \in L_i$ **do**
      - **if** $j \in \mathcal{A}$ **then**
        - $\text{influence}_i \leftarrow \text{influence}_i + \omega_{ji}$;
      - **end**
    - **end**
    - **if** $\mathit{influence}_i \geq \theta_i$ **then** $\mathcal{A} \leftarrow \mathcal{A} \cup \{i\}$;
  - **end**
- **end**
- **return** $\langle \mathit{commit}(|\mathcal{A}|), \phi \rangle$

Protocol 3 establishes the baseline for influence spread computation with compact lists. It implements LT diffusion through repeated activation rounds: each inactive node becomes active when the accumulated influence from active parents reaches its threshold. The protocol first commits to the network structure and thresholds, ensuring that the evaluation refers to a fixed input, and initializes the active set with S.

The simulation reserves N rounds, a worst-case bound because activation is irreversible and every nonterminal round adds at least one node. In each round, the protocol processes all N nodes and sums the weights of their active parents using L_i. Nodes whose accumulated influence reaches θ_i join the active set and can contribute to further activations. The protocol finally returns a commitment to the total number of activated nodes together with the network commitment. Its worst-case cost is $O(N^2 d_{\max})$: each of N rounds scans N lists of padded length $d_{\max}$. A fixed-size circuit for this baseline reserves all these operations, including rounds after diffusion would already have terminated. Compact lists thus improve on the dense baseline in Appendix A.3, but repeated diffusion rounds still make large instances expensive to prove.

B.2.2 Topological Optimization via Acyclicity

Information diffusion with a temporal ordering motivates considering acyclic network representations. Here acyclicity is an explicit input assumption, as specified in Section 3; the optimization applies to the committed DAG. Before committing the network or answering seed-selection queries, the prover computes a topological ordering off-circuit and reindexes the incoming lists, weights, and thresholds consistently. Every real edge (j, i) then satisfies $j < i$. This preprocessing depends only on the network and can be reused across queries.

The ordering enables a single-pass influence computation. When node i is processed, all its parents have already been processed, so their activation states are final. Testing i once therefore determines whether it belongs to the LT closure. The optimized algorithm retains the accumulation and threshold test of Protocol 3, but scans nodes once in topological order instead of repeating the outer diffusion loop. This removes a factor of N, reducing one evaluation from $O(N^2 d_{\max})$ to $O(N d_{\max})$. Consequently, evaluating all candidates in k greedy rounds costs $O(kN^2 d_{\max})$ before pruning and parallelism.

Although sorting is performed off-circuit, the order is not accepted merely on the prover's assertion. The Commitment relation checks $j < i$ for every real incoming edge, together with padding, index

ranges, value ranges, and the no-wrap bound. All evaluation proofs are bound to that validated encoding. This ensures that the single-pass computation has the same closure as the repeated-round baseline on the committed DAG. If a cyclic source graph is converted to a DAG, as in Section 5, that conversion occurs before commitment and the proof concerns the resulting instance.

Modeling boundary and extensions. The DAG restriction is a deliberate tractability boundary of the current system; generic social networks may contain cycles. Removing edges to obtain a DAG trades modeling fidelity for a single topological pass, as quantified in Appendix F.2. A direct cyclic-LT circuit reserves up to N propagation rounds. A possible extension would instead expose a public round bound R and prove R rounds on the cyclic graph, with $O(RNd_{\max})$ work per candidate. Such a proof certifies the bounded-round spread; claiming the final closure additionally requires checking convergence or using a sufficient worst-case round bound.

Independent Cascade (IC) diffusion could likewise be made deterministic inside the relation by committing to the randomness governing edge activation attempts. Each proof would bind the graph, random choices, and resulting diffusion trace. Commitment alone does not ensure unbiased sampling; a sampling protocol would also need to prevent the prover from selecting favorable randomness. Certifying an estimate of expected spread would require proving many sampled traces and their average, increasing circuit and proving costs with the number of samples. These extensions require new relations and are not implemented in the present evaluation.

Algorithm 4 describes witness generation for a full candidate evaluation. The circuit constrains the same operations, including Boolean activation bits, bounded accumulators, integer threshold comparisons, the activation count, and the result commitment.

Algorithm 4: One-pass LT evaluation for candidate v

Input: Seed set $S^{(t-1)}$, candidate v, private (L, θ)
Output: Activation vector $a_{t,v}$ and score $\sigma_{t,v}$

Initialize $a_i \leftarrow 1$ iff $i \in S^{(t-1)} \cup \{v\}$;
for $i \leftarrow 0$ **to** $N-1$ **do**
  $z_i \leftarrow \sum_{(j,\omega_{ji}) \in L_i} \omega_{ji} a_j$;
  $a_i \leftarrow a_i \vee [z_i \geq \theta_i]$;
end
$a_{t,v} \leftarrow (a_0, \ldots, a_{N-1})$; $\sigma_{t,v} \leftarrow \sum_i a_i$;
return $a_{t,v}, \sigma_{t,v}$;

The proof circuit does not interpret arbitrary field elements as ordered values. It range-checks each weight, threshold, accumulator, and score and uses the comparison gadget defined in Appendix C. Consequently, Algorithm 4 and the circuit have the same bounded-integer semantics.

B.2.3 Strategic Pruning via Activation Analysis

The greedy computation admits two further reductions based on activation history. First, if v is already a seed, adding it again leaves the seed set unchanged and therefore has zero marginal gain. Previously selected seeds need no new spread evaluation. Second, even if v is not a seed, it may already belong to the diffusion closure of the current seed set. Seeding such an active node also leaves the closure unchanged, so its marginal gain is zero. These observations allow full spread evaluations to be replaced by cheaper activation-membership checks.

However, visibly skipping these candidates would reveal which nodes had already become active. Pruning must therefore authenticate activation without disclosing the private association between candidate identities and activation states through the proof contents.

For round t, the previous selection proof publishes or commits to $\rho_{t-1} = \mathsf{MT.Commit}(a^{(t-1)})$. If $a_v^{(t-1)} = 1$, equivalently $v \in \mathcal{A}(S^{(t-1)}; L, \theta)$, Equation 8 implies

$$\sigma(S^{(t-1)} \cup \{v\}; L, \theta) = \sigma(S^{(t-1)}; L, \theta), \tag{10}$$

so the candidate has zero marginal gain. PruneEval proves the premise by verifying a leaf value of one under ρ_{t-1}; it cannot be used for a nonmember. Full evaluation is used for every remaining slot.

B.2.4 Privacy-Preserving Pruning via Random Permutation

To obscure the association between candidate slots and node identities, the prover uses a permutation

$$\mathsf{perm} : [N] \to [N], \tag{11}$$

where slot j represents the actual node $v = \mathsf{perm}[j]$. The activation vector remains bound by its Merkle root ρ_{t-1}. For an active candidate, a private authentication path proves that the leaf for v has value one without revealing the complete activation vector. Protocol 5 shows this membership check; Appendix C gives the full PruneEval interface and its query, round, and state bindings.

Protocol 5: Privacy-preserving activation verification

Input: Public slot j, roots ρ_{t-1} and ϕ_{perm}; private perm, its commitment opening, and Merkle path μ_v
Output: An activation-membership assertion for slot j

```
assert the opening of φ_perm is valid for perm;
v ← perm[j];
assert MT.Verify(ρ_{t−1}, v, 1, μ_v);
return true;
```

The prover accesses candidates through $v = \mathsf{perm}[j]$. PermGen derives Fisher–Yates swap indices from a domain-separated transcript and fresh private salt, constrains every swap, and commits to the resulting table. Permutation-aware evaluation and selection circuits open the same ϕ_{perm}. This prevents proof contents from linking slot j directly to node v, subject to the public-input and metadata boundary in Section 3. Appendix E proves the shuffle claim.

Each activation-membership check uses $O(\log N)$ hashes. Applying such checks across N candidate slots incurs up to $O(N \log N)$ hashing work per greedy round, in addition to the permutation and activation-root construction. This overhead can be worthwhile because PruneEval is much cheaper than a full influence evaluation. When many nodes are already active, eliminating their redundant evaluations outweighs the extra permutation and Merkle-tree operations. When few nodes are active, full evaluation can be more efficient. The prover can choose between these evaluation strategies while still providing a valid proof for every candidate slot. As in Section 3, the zero-knowledge guarantee covers proof contents; hiding separately observable circuit choices, pruning counts, or execution timing requires an oblivious, padded deployment interface.

B.3 Implementation Notes

We implement ZINK in Noir with the UltraHonk backend. Poseidon2 (Grassi et al., 2023) instantiates commitments, Merkle hashing, and transcript hashing. Candidate-evaluation leaves within a greedy round are independent, while selection proofs impose the round-to-round dependency. The recursion tree therefore aggregates each level in parallel. The complete circuit interfaces and canonical leaf layout appear in Appendix C; recursion costs appear in Appendix D.

C The ZINK Relation and Its Security Properties

This section defines the computation certified by ZINK and proves soundness, reliability, and zero-knowledge of the cryptographic proof contents.

C.1 Notation and finite-field semantics

Let $[N] = \{0, \ldots, N-1\}$ and let the compact, topologically ordered network representation be (L, θ), where L_i contains at most $d_{\max}$ weighted incoming edges of node i. For a seed set S, let

$$\mathcal{A}(S; L, \theta) \subseteq [N] \quad \text{and} \quad \sigma(S; L, \theta) = |\mathcal{A}(S; L, \theta)| \tag{12}$$

denote the deterministic Linear Threshold (LT) diffusion closure and its influence spread. For greedy round t, write $\mathcal{A}^{(t)} = \mathcal{A}(S^{(t)}; L, \theta)$ and let $a^{(t)} \in \{0,1\}^N$ be its indicator vector. For candidate slot j, $a_{t,j} \in \{0,1\}^N$ similarly denotes the indicator vector of $\mathcal{A}(S^{(t-1)} \cup \{v\}; L, \theta)$ for that slot's

candidate v. We assume the graph is a DAG, so the circuit computes the closure by one pass in the committed topological order. The closure is monotone and idempotent:

$$S \subseteq \mathcal{A}(S; L, \theta), \qquad \mathcal{A}(\mathcal{A}(S; L, \theta); L, \theta) = \mathcal{A}(S; L, \theta). \tag{13}$$

All arithmetic constraints are over the proof field $\mathbb{F}_p$. Comparisons are comparisons of canonical integer representatives, not of unordered field elements. Let ℓ be a public range parameter satisfying $2^{\ell+1} < p$. A range check for $x \in \mathbb{F}_p$ enforces

$$x = \sum_{r=0}^{\ell-1} 2^r x_r \pmod{p}, \qquad x_r(x_r - 1) = 0, \tag{14}$$

and therefore fixes the canonical representative $\bar{x} \in [0, 2^\ell)$. For range-checked x, y, the comparison gadget enforces $x \geq y$ by introducing a range-checked difference u such that

$$x = y + u \pmod{p}. \tag{15}$$

Because $2^{\ell+1} < p$, Equation 15 is satisfiable exactly when $\bar{x} \geq \bar{y}$ as integers. The relation range-checks every weight, threshold, accumulated influence, and score, and fixes public bounds so that every valid sum is below 2^ℓ.

We write $\phi_{\mathsf{net}} = \mathsf{commit}(\langle L, \theta \rangle)$ for the network commitment and ϕ_{perm} for the permutation commitment. The security parameter is λ, which enters setup as $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and bounds all negligible failure probabilities.

C.2 Global statement, witness, and proof view

We define the public configuration as

$$\mathsf{cfg} = (N, k, d_{\max}, w, \ell, \tau) \tag{16}$$

The recursion width is w. The final public statement is

$$s = (\mathsf{qid}, \mathsf{cfg}, \phi_{\mathsf{net}}, S^{(k)}, \sigma_{\mathsf{final}}). \tag{17}$$

Let r_{perm} denote the private permutation salt and perm the resulting permutation. For round t and canonical slot $j \in [N]$, let $\sigma_{t,j}$ be the candidate score, $c_{t,j}$ the full-evaluation/pruning selector, $\mu_{t,j}$ a Merkle authentication path, $\phi_{t,j}$ the slot-result commitment, and $\rho_{t,j}$ a slot activation root when one is produced. Let ρ_t be the activation root of greedy state t. Each (s_i, π_i) denotes a local statement/proof pair, and M_{tree} is the number of real plus padded leaf positions in the recursion tree. The witness is

$$\begin{aligned} W = (L, \theta, r_{\mathsf{perm}}, \mathsf{perm}, \{v_t\}_{t=1}^k, \{S^{(t)}, a^{(t)}, \rho_t\}_{t=0}^k, \\ \{\sigma_{t,j}, c_{t,j}, \mu_{t,j}, \rho_{t,j}, \phi_{t,j}\}_{\substack{1 \leq t \leq k, \\ j \in [N]}}, \{s_i, \pi_i\}_{i=1}^{M_{\mathsf{tree}}}). \end{aligned} \tag{18}$$

It contains the network opening, permutation opening, ordered greedy choices, intermediate states, candidate data, commitment openings, Merkle data, local statements, and recursive child proofs. Let X_{pub} denote the ordered collection of public statements and circuit identifiers of any subproofs exposed outside recursion. If only the final recursive proof is released, then $X_{\mathsf{pub}} = \varnothing$.

Definition C.1 (Cryptographic proof view)**.** *The proof-content view is*

$$\mathsf{View}_{\mathsf{proof}}(\mathsf{pp}, s, X_{\mathsf{pub}}, W) = (\mathsf{pp}, s, X_{\mathsf{pub}}, \Pi_{\mathsf{visible}}, \pi_{\mathsf{final}}), \tag{19}$$

where Π_{visible} *contains any subproof objects released to the verifier.*

C.3 Circuit interfaces

Each circuit is specified by its private witness, local public input, output, computation function, witness generation, and constraint relation. Commitments to local values are domain-separated by (qid, t, j) throughout.

Circuit Commitment
Private witness: L, θ, the initial activation vector $a^{(0)}$, and commitment randomness.
Public input: qid, N, $d_{\max}$, and the declared range bounds.
Output: Network commitment ϕ_{net} and the initial-state commitment.
Computation function: Compute $\phi_{\mathsf{net}} = \mathsf{commit}(\langle L, \theta \rangle)$ and $a^{(0)} = a(\varnothing; L, \theta)$; hash-bind the initial state, including its activation root ρ_0 and spread, alongside the network commitment.
Witness generation: Pad each L_i to length $d_{\max}$ and encode all weights and thresholds in $\mathbb{F}_p$.
Constraint relation: Check the commitments, padding syntax, node-index ranges, topological condition, field-value ranges, and public no-wrap bound; enforce $S^{(0)} = \varnothing$, its LT activation vector, and the corresponding initial-state hash.

Figure 4: Protocol for the Commitment circuit.

Circuit PermGen
Private witness: Fresh salt $r_{\mathsf{perm}} \in \{0,1\}^{\lambda}$, swap indices j_i, and table perm.
Public input: The domain-separated transcript T, including $(\mathsf{qid}, N, \phi_{\mathsf{net}})$ and a circuit digest.
Output: Permutation commitment ϕ_{perm}.
Computation function: Derive j_i from the random oracle and apply the Fisher–Yates swaps to the identity table.
Witness generation: Sample r_{perm} uniformly and compute the swap sequence off circuit.
Constraint relation: Recompute every hash, reduction, and swap; enforce $0 \le j_i \le i$; and check ϕ_{perm}. Section E proves the resulting distribution and consistency properties.

Figure 5: Protocol for the PermGen circuit.

Circuit FullEval
Private witness: L, θ, $S^{(t-1)}$, candidate $v = j$, activation vector $a_{t,j}$, score $\sigma_{t,j}$, and the network, state, and result commitment openings.
Local public input: qid, (t, j), ϕ_{net}, the preceding state commitment, and result commitment $\phi_{t,j}$.
Output: A commitment $\phi_{t,j}$ to $(v, \sigma_{t,j}, a_{t,j})$.
Computation function: Compute $\mathcal{A}(S^{(t-1)} \cup \{j\}; L, \theta)$, its indicator vector $a_{t,j}$, and score $\sigma_{t,j} = \sum_i a_{t,j,i}$.
Witness generation: Run the deterministic topological LT pass for candidate j and commit to the candidate, score, and resulting activation vector.
Constraint relation: Open the common network and preceding-state commitments; enforce $v = j$; constrain every LT accumulation and range-based threshold comparison; count the activation bits; and check $\phi_{t,j}$.

Figure 6: Protocol for the FullEval circuit.

Circuit FullEval+RP
Private witness: L, θ, $S^{(t-1)}$, permutation perm, its commitment opening, candidate $v = \mathsf{perm}[j]$, activation vector $a_{t,j}$, score $\sigma_{t,j}$, and the remaining commitment openings.
Local public input: qid, (t, j), ϕ_{net}, ϕ_{perm}, the preceding state commitment, and result commitment $\phi_{t,j}$.
Output: A commitment $\phi_{t,j}$ to $(v, \sigma_{t,j}, a_{t,j})$.
Computation function: Resolve $v = \mathsf{perm}[j]$, then compute $\mathcal{A}(S^{(t-1)} \cup \{v\}; L, \theta)$, its indicator vector $a_{t,j}$, and score $\sigma_{t,j} = \sum_i a_{t,j,i}$.
Witness generation: Open ϕ_{perm}, read $v = \mathsf{perm}[j]$, run the deterministic topological LT pass for v, and commit to the candidate, score, and resulting activation vector.
Constraint relation: Open the common network, permutation, and preceding-state commitments; enforce $v = \mathsf{perm}[j]$; constrain every LT accumulation and range-based threshold comparison; count the activation bits; and check $\phi_{t,j}$.

Figure 7: Protocol for the FullEval+RP circuit.

Circuit FullEval+RP+MT
Private witness: L, θ, $S^{(t-1)}$, permutation perm, its commitment opening, candidate $v = \mathsf{perm}[j]$, activation vector $a_{t,j}$, score $\sigma_{t,j}$, the Merkle tree for $a_{t,j}$, and the remaining commitment openings.
Local public input: qid, (t, j), ϕ_{net}, ϕ_{perm}, the preceding state commitment, result commitment $\phi_{t,j}$, and activation root $\rho_{t,j}$.
Output: A commitment $\phi_{t,j}$ to $(v, \sigma_{t,j}, a_{t,j})$ and $\rho_{t,j} = \mathsf{MT.Commit}(a_{t,j})$.
Computation function: Resolve $v = \mathsf{perm}[j]$, compute $\mathcal{A}(S^{(t-1)} \cup \{v\}; L, \theta)$, its indicator vector $a_{t,j}$, score $\sigma_{t,j} = \sum_i a_{t,j,i}$, and the Merkle root of $a_{t,j}$.
Witness generation: Generate the same LT witness as FullEval+RP, then construct the Merkle tree whose leaves are the resulting activation bits.
Constraint relation: Enforce all FullEval+RP constraints and recompute the Merkle tree from the constrained activation vector to check $\rho_{t,j}$.

Figure 8: Protocol for the FullEval+RP+MT circuit.

Circuit PruneEval
Private witness: perm, candidate $v = \mathsf{perm}[j]$, the preceding activation vector $a^{(t-1)}$, Merkle path $\mu_{t,j}$, and commitment openings.
Local public input: qid, (t, j), ϕ_{net}, ϕ_{perm}, the preceding state commitment, and the slot-output commitment.
Output: A slot commitment $\phi_{t,j}$ encoding score $\sigma(S^{(t-1)}; L, \theta)$, hence marginal gain zero.
Computation function: Check that $a_v^{(t-1)} = 1$.
Witness generation: Open the leaf for v in the Merkle commitment $\rho_{t-1} = \mathsf{MT.Commit}(a^{(t-1)})$.
Constraint relation: Check the permutation opening, the state/root linkage, and $\mathsf{MT.Verify}(\rho_{t-1}, v, 1, \mu_{t,j}) = 1$. By idempotence in Equation 13, enforce the same score as a full evaluation of $S^{(t-1)} \cup \{v\}$.

Figure 9: Protocol for the PruneEval circuit. Zero-knowledge hides its private witness but is conditioned on the circuit's declared local public input.

Circuit family Select and Select+RP
Private witness: Openings of all N slot commitments, their scores, candidates, and the previous and next greedy states.
Local public input: qid, round t, ϕ_{net}, the preceding state commitment, and the ordered vector $(\phi_{t,0}, \ldots, \phi_{t,N-1})$.
Output: Selected node v_t, state $S^{(t)} = S^{(t-1)} \cup \{v_t\}$, spread $\sigma(S^{(t)}; L, \theta)$, and a commitment to the next state and activation root.
Computation function: Select the maximum marginal gain, using the public deterministic tie rule τ. The RP variant maps the selected slot through perm.
Witness generation: Open all score commitments and compute the argmax.
Constraint relation: Check every opening, distinct-seed condition, $N - 1$ range-based comparisons, tie handling under τ, and consistency of the selected slot with the next state.

Figure 10: Protocol for the Select and Select+RP circuit family.

Circuit ProofRecursion
Private witness: Child proofs π_i, their complete public-input vectors s_i, circuit tags, intermediate commitments, and dummy proofs for unused tree positions.
Public input: The state inputs and outputs certified by the aggregated subexecution; at the root, the final statement s from Equation 17. Private states are represented by their commitments.
Output: One recursive proof π_{final}.
Computation function: Aggregate a fixed-width tree of commitment, permutation, candidate-evaluation, and selection proofs.
Witness generation: Arrange every real leaf at its canonical position and fill only positions beyond the real leaf count with the dedicated dummy relation.
Constraint relation: Verify each child proof against its complete public input; check circuit tags, one indexing mode per round, and coverage of all (t, j); enforce common ϕ_{net} and ϕ_{perm}; match the first round's input state to the certified initial state and each later input state to its predecessor's output state through these public inputs; and open the final state to $(S^{(k)}, \sigma_{\mathsf{final}})$.

Figure 11: Protocol for the ProofRecursion circuit and statement chaining.

Table 3: ZINK circuits and their role in the composed relation ($N = 1000$ constraint counts).

Circuit	Certified relation	Constraints
Commitment	Binds the compact network and thresholds	$\sim$ 280K
PermGen	Binds a Fisher–Yates permutation	$\sim$ 1034K
FullEval	Evaluates one unpermuted candidate	$\sim$ 2516K
FullEval+RP	Evaluates one permuted candidate	$\sim$ 2546K
FullEval+RP+MT	Also commits to the resulting activation vector	$\sim$ 11229K
PruneEval	Proves zero marginal gain by activation membership	$\sim$ 36K
Select	Proves the unpermuted greedy argmax	$\sim$ 118K
Select+RP	Proves the permuted greedy argmax	$\sim$ 297K
ProofRecursion	Aggregates child proofs and checks statement chaining	$\sim$ 2252K

C.4 COMPOSED PROTOCOL

Protocol 6 makes the security-critical ordering explicit. The root ρ_{t-1} used to prune round t commits to the closure of the already selected set $S^{(t-1)}$; a root produced by evaluating a new candidate is used only after that candidate is selected.

Algorithm 6: ZINK protocol

Input: Public $(\mathsf{qid}, \mathsf{cfg})$; private (L, θ)
Output: s and π_{final}

$S^{(0)} \leftarrow \varnothing$; $a^{(0)} \leftarrow a(S^{(0)}; L, \theta)$; $\rho_0 \leftarrow \mathsf{MT.Commit}(a^{(0)})$;
Run Commitment on (L, θ) to certify ϕ_{net} and the initial-state commitment in one proof;
$(\mathsf{perm}, \phi_{\mathsf{perm}}) \leftarrow \mathsf{PermGen}(\mathsf{qid}, N, \phi_{\mathsf{net}})$; generate a PermGen proof;
for $t \leftarrow 1$ **to** k **do**
    Choose one indexing mode and its matching selection circuit for the entire round;
    for $j \leftarrow 0$ **to** $N-1$ **do**
        Use PruneEval only in permuted mode with a witness for $a^{(t-1)}_{\mathsf{perm}[j]} = 1$; otherwise use the full-evaluation circuit for the chosen mode;
        Generate one candidate-evaluation proof and result commitment $\phi_{t,j}$;
    end
    $(v_t, S^{(t)}, a^{(t)}, \rho_t) \leftarrow \mathsf{Select(+RP)}(\phi_{t,0}, \ldots, \phi_{t,N-1})$;
    Generate the selection proof and bind its input state to round $t-1$ and its output state to round t;
end
$\sigma_{\mathsf{final}} \leftarrow \sum_i a_i^{(k)}$; form s as in Equation 17;
Place every real subproof at its canonical leaf; fill only surplus tree positions with valid dummy proofs;
$\pi_{\mathsf{final}} \leftarrow \mathsf{ProofRecursion}(s, W)$ using public width w; output $(s, \pi_{\mathsf{final}})$;

With one candidate-evaluation proof and one selection proof per round, the number of real leaves is $M_{\mathsf{real}} = 2 + k(N+1)$. Recursion checks a tagged canonical position for every real leaf, so a dummy proof cannot replace an omitted evaluation or selection.

C.5 SECURITY ASSUMPTIONS

We assume a few basic points: (i) the underlying recursive NIZK is complete, knowledge-sound, and zero-knowledge; (ii) all commitments used for private values are binding and hiding at security level λ; (iii) the Merkle and transcript hashes are collision resistant, and the Fiat–Shamir hash is a random oracle where invoked.

C.6 SECURITY THEOREMS

Theorem C.1 (Soundness)**.** *Except with probability* $\mathsf{negl}(\lambda)$*, if* $\mathsf{Verify}(\mathsf{pp}, s, \pi_{\mathsf{final}}) = 1$*, then there exist* (L, θ) *opening* ϕ_{net} *and an ordered sequence* $(v_1, \ldots, v_k)$ *such that each* v_t *is selected by the declared greedy rule* τ *on* (L, θ)*,* $S^{(k)} = \{v_1, \ldots, v_k\}$*, and* $\sigma_{\mathsf{final}} = \sigma(S^{(k)}; L, \theta)$*.*

Proof. Knowledge soundness extracts a satisfying witness for the root relation. Recursive verification and canonical leaf tags yield a valid commitment proof, one valid permutation proof, every required slot proof, and every selection proof. Binding fixes a single (L, θ) and a single perm across all leaves. The commitment proof certifies the initial activation state, and verification against complete public inputs binds each subsequent state transition.

For a full slot, the extracted witness satisfies the deterministic LT constraints and therefore gives the correct $\sigma(S^{(t-1)} \cup \{v\}; L, \theta)$. For a pruned slot, Merkle binding to the certified predecessor state gives $v \in \mathcal{A}(S^{(t-1)}; L, \theta)$. By idempotence,

$$v \in \mathcal{A}(S^{(t-1)}; L, \theta) \Longrightarrow \sigma(S^{(t-1)} \cup \{v\}; L, \theta) = \sigma(S^{(t-1)}; L, \theta), \tag{20}$$

so the pruned score equals the full-evaluation score and its marginal gain is zero. The selection relation opens all N slot commitments, applies sound integer comparisons without wraparound, and enforces τ on ties. State chaining gives $S^{(t)} = S^{(t-1)} \cup \{v_t\}$. Induction over $t = 1, \ldots, k$ establishes the greedy sequence, and the final state opening establishes the claimed spread. □

Definition C.2 (Reliability)**.** *Reliability is completeness plus cross-circuit consistency: an honest execution on a valid committed instance is accepted, and every accepted module is bound to the same query, network, permutation, ordered slot set, and predecessor/successor state.*

Theorem C.2 (Reliability)**.** *For every valid committed DAG and every honest execution of Protocol 6, the verifier accepts with probability* $1 - \mathsf{negl}(\lambda)$*. Pruning, recursion, and dummy padding do not change the resulting seed set or influence spread.*

Proof. Honest full-evaluation witnesses satisfy the LT relation. Whenever pruning is chosen, the honest Merkle path proves membership in the preceding closure, and Equation 20 shows that the logical score is identical to full evaluation. Thus Select(+RP) receives the same candidate scores as an execution without pruning. The public tie rule fixes the same selected node. Every real proof carries the common query and commitments, and the next-state output of round t is the input-state commitment of round $t+1$. Dummy proofs occur only beyond the tagged real-leaf range and cannot affect a state transition. Completeness at the leaves and induction up the fixed recursion tree give acceptance except for negligible completeness error. □

Theorem C.3 (Zero-knowledge of proof contents)**.** *For every fixed final statement* s *and exposed local public-input collection* X_{pub}*, there is a probabilistic polynomial-time simulator* $\mathcal{S}$ *such that*

$$\mathcal{S}(\mathsf{pp}, s, X_{\mathsf{pub}}) \approx_c \mathsf{View}_{\mathsf{proof}}(\mathsf{pp}, s, X_{\mathsf{pub}}, W). \tag{21}$$

Consequently, the cryptographic proof objects reveal no information about their private witnesses beyond what is already contained in s *and* X_{pub}*.*

Proof. For each visible subproof, apply zero-knowledge of the underlying NIZK to replace the real proof with a simulated proof for its fixed local public statement in X_{pub}. Apply zero-knowledge of the recursive NIZK to replace π_{final} with a simulated proof for s. A standard hybrid over the polynomially many visible proof objects changes the verifier's view by at most a negligible sum of zero-knowledge advantages. The simulator makes no claim that X_{pub} or external execution metadata is independent of the witness. □

C.7 Aggregate pruning leakage

Permutation conceals the association between candidate slots and node identities; it does not conceal how many slots are pruned. Let

$$p_t = \sum_{j=0}^{N-1} \mathbf{1}[c_{t,j} = \mathsf{prune}], \qquad \mathcal{L}_{\mathsf{prune}}(s, X_{\mathsf{pub}}, W) = \big(s, X_{\mathsf{pub}}, (p_t)_{t=1}^{k}\big). \tag{22}$$

For an interface that reveals per-round pruning counts, this is declared aggregate leakage accepted in exchange for cheaper candidate proofs. If every active candidate is pruned, $p_t = |\mathcal{A}^{(t-1)}|$; otherwise p_t counts only the candidates actually processed by PruneEval. The hidden slot-to-node mapping, edges, weights, and thresholds remain private witnesses, subject to inferences from the declared public outputs and leakage. In particular, publicly returned seed identities are not claimed to be secret.

The intended interface releases only the recursive root proof and its public statement. Theorem C.3 guarantees that proof objects reveal nothing further beyond their public statements, including the public inputs of any released subproofs. This guarantee does not hide separately observable circuit choices, pruning counts, scheduling, traffic, or timing, nor does it assert that arbitrary execution metadata is determined by Equation 22. Circuit identifiers in released subproofs belong to X_{pub}; metadata observed outside these statements must also be accounted for in a deployment's leakage profile. Restricting this metadata to counts, or hiding the counts as well, requires an appropriate oblivious, padded interface.

Remark (All nodes activated)**.** *If $|\mathcal{A}^{(t-1)}| = N$ and this cardinality is disclosed, the observer learns that every node is active. Since all nodes share the same activation state, there is no additional ambiguity about individual activation identities for a permutation to protect. Permuting the all-one activation vector does not change it. This does not invalidate Theorem C.3, because that theorem concerns only what the cryptographic proof objects reveal beyond their public statements. It does mean that the permutation provides no additional protection for separately observable circuit-selection or execution metadata.*

D Proof-Aggregation Complexity

D.1 Aggregation tree

Recursive aggregation replaces a large batch of application proofs with one root proof by repeatedly verifying several child proofs inside a parent circuit. Let M be the number of leaf proofs, let w_r be the fanout (number of children per parent) at aggregation level r, and write $\mathbf{w} = (w_1, w_2, \ldots)$ for the fanout sequence. For the complete ZINK relation before padding, $M = M_{\mathsf{real}} = 2 + k(N+1)$ as defined in Equation 6. Define

$$n_0 = M, \qquad n_r = \left\lceil \frac{n_{r-1}}{w_r} \right\rceil, \qquad D = \min\{r : n_r = 1\}. \tag{23}$$

The number of newly generated aggregation proofs is

$$A(M, \mathbf{w}) = \sum_{r=1}^{D} n_r. \tag{24}$$

Every non-root proof is verified once by its parent. Hence the exact number of in-circuit child-proof verifications is

$$E(M, \mathbf{w}) = \sum_{r=1}^{D} n_{r-1} = M + A(M, \mathbf{w}) - 1. \tag{25}$$

For constant width w, $D = \lceil \log_w M \rceil$, $A = \Theta(M/(w-1))$, and $E = \Theta(M)$.

Let q_{r-1} be the number of fields in a level-$(r-1)$ child statement, and let $C_r(w_r, q_{r-1})$ be the constraint cost of one level-r aggregation circuit, including its w_r child verifiers and statement reconstruction. The prover's aggregation work is

$$W_{\mathsf{agg}} = \sum_{r=1}^{D} n_r C_r(w_r, q_{r-1}). \tag{26}$$

With fixed backend formats, fixed w, and bounded q_r, this is $\Theta(M)$. Let $T_r(w_r, q_{r-1})$ be the wall time for one level-r aggregation proof. If all nodes at a level are proved in parallel, the critical path is

$$T_{\mathsf{crit}} = \sum_{r=1}^{D} T_r(w_r, q_{r-1}), \tag{27}$$

which is logarithmic in M when the per-level time is bounded. A larger w shortens the tree but increases each aggregation circuit's constraints and memory; a smaller w reduces per-circuit memory but creates more proofs and levels.

D.2 Verifier cost and succinctness

The verifier receives one root proof and invokes the native verifier once. Let $B_{\pi,i}$ and $B_{x,i}$ be the sizes of leaf i's proof and public statement, and let $B_{\pi,\mathsf{root}}$ and $B_{x,\mathsf{root}}$ be the corresponding root sizes. Then

$$B_{\mathsf{direct}} = \Theta\left(\sum_{i=1}^{M} (B_{\pi,i} + B_{x,i})\right), \qquad B_{\mathsf{recursive}} = B_{\pi,\mathsf{root}} + B_{x,\mathsf{root}}. \tag{28}$$

Succinctness makes $B_{\pi,\mathsf{root}} = O(1)$ with respect to M. ZINK carries the shared network and intermediate states through commitments and exposes only the declared root statement, while the recursion relation checks the common commitments and state chaining described in Fig. 11.

For an illustrative candidate-proof batch with $M = 5,000$ leaves, a fixed-width $w = 10$ aggregation trace has

$$5000 \longrightarrow 500 \longrightarrow 50 \longrightarrow 5 \longrightarrow 1, \tag{29}$$

so it generates 556 aggregation proofs, performs 5,555 in-circuit child verifications (5,560 with padding), and has four dependent levels.

D.3 Overall measured tradeoff

Table 4 reports three representative points at $k = 5$. The added proving time is the cost of recursion relative to proving the unaggregated subproofs, and the other columns report the resulting verifier and proof-size improvements.

Table 4: Representative end-to-end aggregation results ($k = 5$).

N	Added proving time	Verifier speedup	Proof-size reduction
100	36.59%	4.2×	250.9×
1000	29.35%	83.1×	2473.5×
5000	10.75%	1187.5×	12325.8×

At $N = 5000$, aggregation adds 10.75% to the reported proving time while reducing external verification from 890.63 seconds to 0.75 seconds and proof size from 169.48 MB to 13.75 KB. The percentage overhead is not monotone in N because the circuit mix, recursion width, parallel utilization, and memory pressure also change.

D.4 Scaling to larger networks

The current size limit is driven by prover memory and proof-generation cost in the ZKP backend. Independent candidate proofs already permit multi-machine execution, yielding the measured 7.68× speedup on eight machines in Section 5.3, while the specialized circuits reduce proving time by up to 42.72× relative to the evaluated zkVMs. Further scaling could combine hardware acceleration of the backend's arithmetic and cryptographic kernels with recursion backends that reduce child-verifier constraints and peak memory. The fanout tradeoff above remains relevant: a wider tree reduces dependent levels but raises the memory cost of each aggregation circuit. These are directions for future implementation and benchmarking, rather than measured improvements in the current system.

E Fiat–Shamir Randomness in PermGen

This section formalizes the hash-derived randomness used by the Fisher–Yates shuffle.

E.1 Construction

Let $[N] = \{0, \ldots, N-1\}$ and let $\mathbb{F}_p$ be the proof field. Let $H_{\mathsf{perm}} : \{0,1\}^* \to \mathbb{F}_p$ be the domain-separated random-oracle hash used to expand permutation randomness, and let H_{com} be the domain-separated hiding and binding commitment hash. The security parameter λ appears in setup, in the length of the fresh permutation salt, and in all negligible error terms. Let Encode be a canonical, injective, length-delimited encoding, version the protocol-version string, and $\mathsf{circuit_digest}$ the digest of the constrained circuit. The public transcript is

$$T = \mathsf{Encode}(\mathsf{version}, \mathsf{circuit_digest}, N, \phi_{\mathsf{net}}, \mathsf{qid}). \tag{30}$$

An honest prover samples

$$r_{\mathsf{perm}} \leftarrow \{0,1\}^{\lambda} \tag{31}$$

uniformly and independently for this query. For $i = N-1, \dots, 1$, it derives

$$z_i = H_{\mathsf{perm}}(\texttt{ZINK/PermGen/v1} \parallel T \parallel \mathsf{Encode}(r_{\mathsf{perm}}, i)) \in \mathbb{F}_p,$$
$$j_i = \bar{z}_i \bmod (i+1), \tag{32}$$

where $\bar{z}_i \in \{0, \dots, p-1\}$ is the canonical integer representative. Starting from $A = [0, 1, \dots, N-1]$, PermGen swaps $A[i]$ and $A[j_i]$ and outputs $\mathsf{perm} = A$. It commits to the result as

$$\phi_{\mathsf{perm}} = H_{\mathsf{com}}(\texttt{ZINK/PermGen/commit/v1} \parallel T \parallel \mathsf{Encode}(r_{\mathsf{perm}}, \mathsf{perm})). \tag{33}$$

The circuit constrains Equation 32, every range reduction and swap, and Equation 33. All RP circuits are bound to the same $(T, \phi_{\mathsf{perm}})$ in their local statements.

E.2 Assumptions

We assume that H_{perm} is a random oracle with output uniform in $\mathbb{F}_p$; the honest prover's salt is fresh, uniform, and hidden from the verifier; Equation 33 is computationally binding and hiding; and the underlying recursive NIZK is complete, knowledge-sound, and zero-knowledge. Domain separation ensures that permutation-expansion queries cannot be confused with commitment or other protocol queries.

Lemma E.1 (Independent hash values)**.** *For fixed (T, r_{perm}), the values $z_{N-1}, \dots, z_1$ in Equation 32 are independent and uniform in $\mathbb{F}_p$ in the random-oracle experiment.*

Proof. The canonical encoding contains a distinct index i in each query, so the oracle inputs are pairwise distinct. A random oracle assigns independent uniform outputs to distinct inputs. □

Lemma E.2 (Modulo-reduction bias)**.** *Let Z be uniform on $\{0, \dots, p-1\}$, let $J = Z \bmod m$, let U_m be uniform on $\{0, \dots, m-1\}$, and write Δ for statistical distance. Then*

$$\Delta(J, U_m) \le \frac{m}{4p}, \tag{34}$$

Proof. Write $p = qm + r$ with $0 \le r < m$. Exactly r residues have $q+1$ preimages and the other $m-r$ residues have q preimages. Direct substitution in the definition of statistical distance gives

$$\Delta(J, U_m) = \frac{r(m-r)}{mp} \le \frac{m}{4p}. \tag{35}$$

□

Lemma E.3 (Uniform Fisher–Yates map)**.** *If each J_i is independently uniform on $\{0, \dots, i\}$, Fisher–Yates outputs every permutation in the symmetric group $\mathfrak{S}_N$ with probability $1/N!$.*

Proof. The choice sequence $(J_{N-1}, \dots, J_1)$ has

$$\prod_{i=1}^{N-1} (i+1) = N! \tag{36}$$

possible values. Each sequence determines one output. Conversely, reversing the swaps recovers the unique sequence for any output permutation. Thus the map from choice sequences to permutations is a bijection. □

Theorem E.4 (Security of PermGen)**.** *Let P_N be the honest-prover permutation distribution defined above and let $U_{\mathfrak{S}_N}$ be uniform over $\mathfrak{S}_N$. Under the stated assumptions:*

1. *The honest-prover distribution satisfies*

$$\Delta(P_N, U_{\mathfrak{S}_N}) \le \sum_{i=1}^{N-1} \frac{i+1}{4p} < \frac{N(N+1)}{8p}. \tag{37}$$

2. *Except with the NIZK knowledge-soundness and commitment-binding errors, every accepted PermGen proof opens to a bijection, and every accepted RP circuit in the final recursive proof uses that same table.*

3. *Conditioned on the public statements of the exposed proof objects, their contents reveal no permutation opening, up to the NIZK zero-knowledge and commitment-hiding advantages.*

Proof. By Lemma E.1, the inputs to the modular reductions are independent. Lemma E.2 bounds each reduced index's distance from its ideal uniform distribution. Lemma E.3 identifies the final ideal distribution with $U_{\mathfrak{S}_N}$.

For validity, the circuit starts from the identity table and performs only swaps with $0 \le j_i \le i$. Every satisfying execution therefore ends in a bijection. Knowledge soundness transfers this fact to accepted proofs. Binding of ϕ_{perm} and equality checks on the common local statement prevent two consuming circuits from opening the commitment to different tables. Finally, apply zero-knowledge to each exposed proof conditioned on its public statement, and the permutation opening and salt remain private witnesses. □

For the implementation regime $N \le 5000$ and $p > 2^{200}$, the statistical term in Equation 37 is below 2^{-178}.

Remark (What Fiat–Shamir does and does not guarantee)**.** *The uniformity claim is an honest-prover claim and explicitly depends on Equation 31. Hashing a prover-chosen salt does not prevent that prover from trying many salts and selecting a favorable permutation. Such grinding does not break the bijection or cross-circuit consistency statements, and it is not needed for soundness of greedy execution. If unbiased randomness against a malicious prover is required, r_{perm} must instead be bound to an external unpredictable beacon, a verifier contribution, or a commit-then-reveal coin-flipping protocol.*

F Additional Experimental Details

This appendix supplements Section 5 with backend versions, SRS provenance, proving-key and proof-size accounting, circuit and memory breakdowns, preprocessing impact, fixed-threshold and Monte Carlo greedy comparisons, detailed network and spread measurements, seed-selection trajectories, ablation measurements, and per-round pruning results.

F.1 Proving-Time and Memory Breakdowns

Table 5 gives the per-circuit times, execution counts, and proportions for the 100-node workload underlying Section 5.2. Figure 13 supplies the peak-memory measurements across network sizes. Individual candidate circuits can be released after their subproofs are generated.

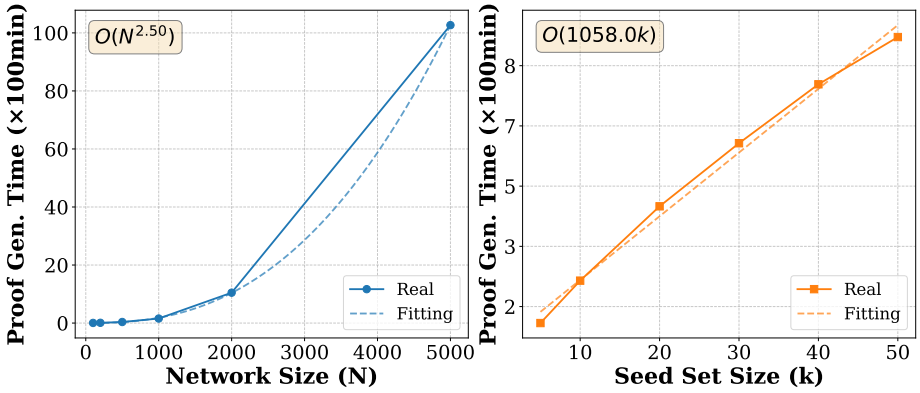


Figure 12: Proof-generation scaling with network size and seed-set size.

Backend versions and SRS provenance. Our implementation pins `@aztec/bb.js` to version `3.0.0-nightly.20251104` and `@noir-lang/noir_js` to `1.0.0-beta.15`; the recursive verifier library `bb_proof_verification` uses the same Barretenberg release tag (Aztec Protocol, 2025b). That release's default SRS distribution is the BN254 Ignition transcript, published

as `g1.dat` and `g2.dat` at https://crs.aztec.network (Aztec Protocol, 2025a; 2020). The ceremony's trust assumption requires at least one participant to have honestly generated and destroyed its secret randomness.

Setup and proving-key generation. UltraHonk uses a one-time universal structured reference string (SRS), as described in Section 3.3. In our evaluated Barretenberg UltraHonk backend, per-circuit preprocessing (proving-key generation) takes 4min40s at $N = 1000$ and 20min27s at $N = 5000$. These are circuit-preprocessing measurements, distinct from the one-time universal setup. We do not report a serialized proving-key size because this backend constructs the prover-side key material internally in memory during proof generation and does not materialize or expose a standalone serialized proving key.

Proof-size accounting. The unaggregated proof sizes in Table 2 measure a gzip-compressed collection of subproofs. SP1 and Risc0 use their own recursive proof pipelines, so the reported sizes and verification times reflect each system's final proof format.

Table 5: Time breakdown of ZINK for 100 node networks.

Circuit	Time	Proportion
FullEval+RP	170.74s	79.5%
FullEval	24.56s	11.5%
PruneEval	9.30s	4.3%
FullEval+RP+MT	5.20s	2.4%
Select+RP	2.72s	1.3%
Select	0.30s	0.1%
Commitment	0.34s	0.2%
PermGen	1.64s	0.8%
Total	**214.8s**	**100.0%**

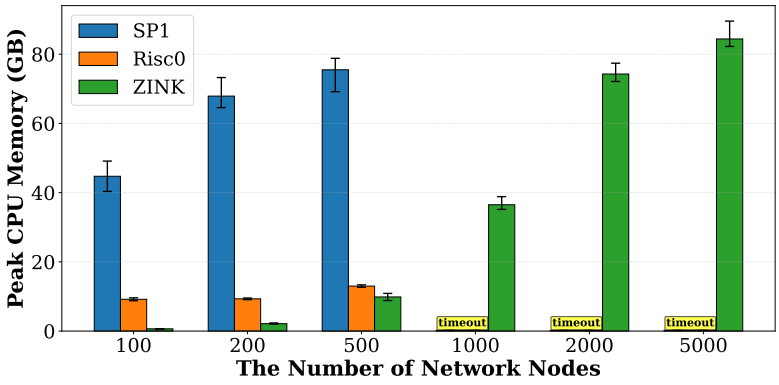


Figure 13: Peak memory consumption during proof generation.

F.2 REAL-NETWORK AND DISTRIBUTED-RUN DETAILS

Email-Eu records email communication within a research institution, Wiki-Vote records Wikipedia administrator votes, and Hamsterster records friendships on a social-networking site (Leskovec et al., 2007a; 2010; Kunegis, 2013). Table 6 supplements the real-network results in Section 5.3 with separate original and processed edge counts. Original counts refer to the source datasets; node counts and processed edges refer to the committed DAGs, including the 5,000-node Wiki-Vote subgraph. Thus, the Wiki-Vote original edge count precedes subgraph selection as well as cycle breaking and degree capping.

Impact of cycle breaking and degree capping. We evaluate the complete preprocessing pipeline on Email-Eu, Hamsterster, and Wiki-Vote. For each dataset, let G be the original cyclic graph, S_{DAG}

Table 6: Performance on real-world networks ($k = 5$), with edge counts before and after preprocessing. Nodes and processed edges describe the proved DAG; original edges describe the source dataset.

Dataset	Nodes	Original edges	Processed edges	Proof gen. (min)	Spread
Wiki-Vote	5,000	103,689	16,022	14,653.88	35.9%
Email-Eu	1,005	25,571	5,338	102.19	76.9%
Hamsterster	2,426	16,631	11,749	1,629.36	74.0%

the seeds selected by greedy on the processed DAG, and S_{orig} the seeds selected by greedy directly on G under the same seed budget. We evaluate both seed sets on G and normalize the spread of S_{DAG} by that of S_{orig}, namely $\sigma_G(S_{\mathsf{DAG}})/\sigma_G(S_{\mathsf{orig}})$. Across the three datasets, the processed-graph seeds retain **90.93%** of the original-graph greedy spread on average. This measures the combined influence loss from preprocessing and quantifies the efficiency–fidelity tradeoff of the committed DAG. The 100% accuracy results in Section 5.4 concern a separate property: proof generation preserves greedy execution on the same processed instance.

F.3 Detailed Spread and Seed-Selection Results

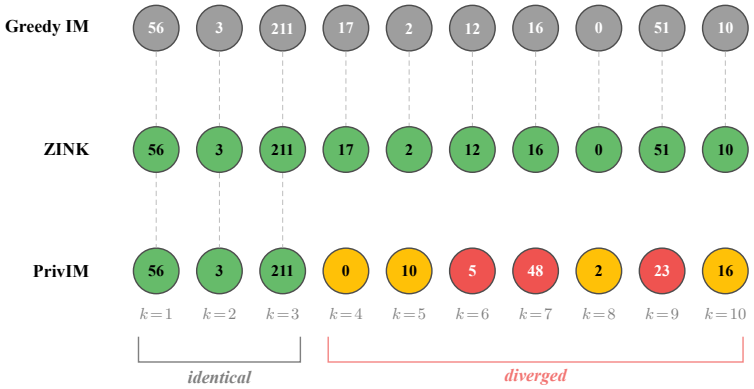
Fixed-threshold versus Monte Carlo greedy. To assess the quality of optimizing one fixed threshold sample, we compare fixed-threshold greedy with Monte Carlo greedy, which estimates expected LT spread over random threshold draws during seed selection. We compare the resulting seed sets by their expected influence. Across evaluated networks with 100–5,000 nodes, fixed-threshold greedy achieves **86–93%** of the expected influence of Monte Carlo greedy on average, while its seed-selection computation is **over** $50\times$ **faster**. This is an empirical quality–runtime comparison; the classical guarantee in Appendix A.2 does not transfer to an arbitrary fixed threshold instance. The speedup concerns greedy computation, while Table 2 separately reports cryptographic proof-generation costs.

Table 7 provides the complete dataset-level activation counts supporting Section 5.4; relative loss is listed for ZINK and PrivIM, in that order. Figure 14 shows the round-by-round seed choices on Synthetic-1000. PrivIM's trajectory diverges after the third round.

Table 7: Influence spread comparison: ZINK vs. baselines.

Dataset	**Plaintext**	**ZINK**	**PrivIM**	**Relative Loss**
Synthetic-500	76	76	51	0% / 32.9%
Synthetic-1000	385	385	208	0% / 46.0%
Wiki-Vote	1793	1793	1156	0% / 35.5%
Email-Eu	773	773	613	0% / 20.7%
Hamsterster	1796	1796	1363	0% / 24.1%

Figure 14: Seed selection convergence: ZINK vs. PrivIM. *Green seeds are correctly selected, yellow seeds are selected in a different order, red seeds are wrongly selected.*



F.4 ADDITIONAL OPTIMIZATION DETAILS

Table 8 reports the ablation measurements underlying Section 5.5 on the 100-node workload with $k = 5$.

Table 8: Ablation on a 100-node network with $k = 5$.

Configuration	Proof Gen.	Constraints	Speedup
Full adjacency matrix	55,356.4s	$\sim$ 21.5B	1.0×
Compact representation	31,523.8s	$\sim$ 11.4B	1.76×
+ topological order	269.4s	$\sim$ 74.7M	205×
+ pruning (full ZINK)	214.8s	$\sim$ 47.0M	258×

Dense-baseline limit. The full-adjacency-matrix variant exhausts memory at $N = 500$, motivating the 100-node comparison in Table 8.

Per-round pruning measurements. Figure 15 supplies the Synthetic-1000 measurements discussed in Section 5.5.

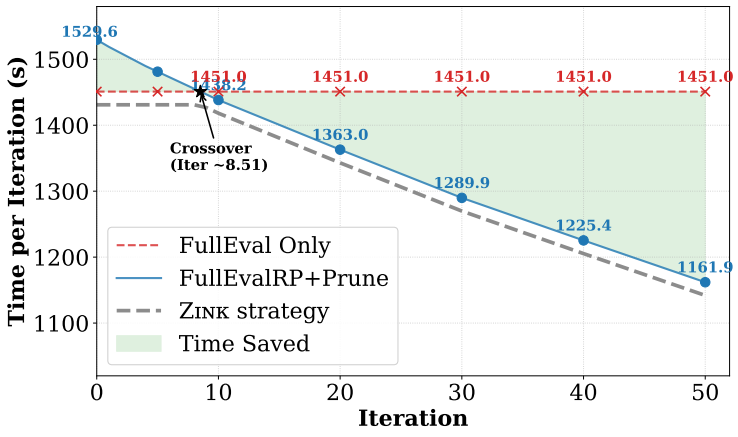


Figure 15: Pruning effectiveness across greedy iterations on Synthetic-1000 dataset.

Arithmetic overhead. The fixed-point implementation requires additional opcodes for overflow checking and bit manipulation. The field-native implementation represents bounded integer weights and thresholds directly in $\mathbb{F}_p$ and uses range checks to preserve their integer semantics.