EchoPrime: A Verifiable Oracle for Deterministic Safe Prime Generation with On-Chain Audit Trails

Darren J. Edwards, Ph.D.*
Founder, Mikoshi Ltd
mikoshiuk@gmail.com

Mikoshi Ltd – EchoPrime
www.echoprime.xyz

February 9, 2026

Abstract

EchoPrime is an Ethereum-compatible oracle and cryptographic primitive for the verifiable publication of deterministic safe primes. Given a symbolic index $n \in \mathbb{N}$, the system projects a candidate safe prime $p = 2q + 1$, evaluates it against a structural verification layer based on Lucas' theorem, confirms primality, and publishes the result on-chain with full trace metadata.

No existing system combines deterministic index-based projection with on-chain auditability and structural verification traces. EchoPrime fills this gap by replacing opaque, entropy-dependent prime generation with a reproducible, publicly auditable pipeline. The system is applicable to trusted setup ceremonies for ZK rollups, group parameter generation in MPC protocols, Diffie–Hellman key exchange, and RSA parameter sourcing.

This paper describes the estimator design, the structural verification mechanism, million-scale empirical validation results, and the on-chain oracle architecture.

1 Introduction

Safe primes p such that $q = \frac{p-1}{2}$ is also prime are foundational to many cryptographic protocols, including zero-knowledge proofs [1], multiparty computation [2], secure key exchange [3], and digital signature schemes [4]. Traditional safe prime generation relies on entropy-heavy random sampling followed by probabilistic primality testing. These methods produce

*Ph.D. in Computational Modelling of Cognition.

correct results but are non-deterministic: given the same inputs, they yield different outputs, and there is no intrinsic record of *why* a particular prime was chosen.

EchoPrime addresses this by introducing a public, Ethereum-compatible oracle for the verifiable publication of safe primes. Rather than relying on random sampling, EchoPrime projects candidate primes from a deterministic estimator function, evaluates them using a structural verification layer, confirms primality, and records the full trace on-chain. Every published prime is deterministically reproducible from its index, accompanied by a numeric verification score, and permanently anchored in an auditable smart contract registry.

The system's core contributions are:

1. **Deterministic index-to-prime projection.** A closed-form estimator maps natural number indices to candidate safe primes, enabling reproducible generation without entropy.

2. **Structural verification via Lucas' theorem.** A scoring layer based on binomial coefficient divisibility provides a deterministic, reproducible primality proxy that produces an auditable numeric trace for each candidate.

3. **On-chain publication with full trace metadata.** Verified primes, their indices, structural scores, and primality verdicts are recorded on an Ethereum-compatible blockchain and emitted as indexable events.

4. **Composable oracle architecture.** The smart contract registry is queryable by other contracts, ceremony tooling, and off-chain systems, enabling integration into ZK rollups, MPC protocols, and trusted setup pipelines.

Tools such as OpenSSL can generate primes deterministically given a seed, and standard libraries provide efficient probabilistic primality testing. EchoPrime's contribution is not in the generation or testing of primes per se, but in combining a deterministic index-based projection pipeline with on-chain auditability and structural verification traces—creating a publicly inspectable record that links each prime to its derivation.

2 Estimator Design

The EchoPrime estimator is a closed-form function that maps a natural number index n to a candidate safe prime p. The estimator targets the structured regions of the number line where safe primes are known to be relatively dense, using the asymptotic distribution of primes and safe primes to calibrate its projection.

2.1 Asymptotic Context

The prime counting function satisfies $\pi(x) \sim x/\ln x$ [10], and the density of safe primes follows $\pi_{\text{safe}}(x) \sim C \cdot x/\ln^2 x$, where C is the Hardy–Littlewood twin prime-like constant for safe primes [5]. The quadratic logarithmic denominator reflects the requirement that both p and $q = (p-1)/2$ must be prime.

2.2 Projection Formula

The estimator computes a candidate from index n as:

$$\hat{p}(n) = 2 \cdot \left\lfloor \alpha \cdot n \cdot \ln^2(n+2) \right\rfloor + 1$$

where α is an empirically calibrated constant that adjusts the projection density, derived from the Bateman–Horn conjecture [6]. The $\ln^2$ factor compensates for the quadratic thinning of safe primes, and the floor-and-odd construction ensures each candidate is odd. The corresponding Sophie Germain candidate is $\hat{q} = (\hat{p} - 1)/2$.

This formula is fully deterministic: the same index always produces the same candidate. There is no randomness, no entropy source, and no dependence on system state. Any party with the formula and the constant α can independently reproduce every candidate.

2.3 Design Philosophy

The estimator does not attempt to enumerate all safe primes. Instead, it projects into regions of the number line where safe primes are statistically likely, producing a subset of candidates that can be verified efficiently. The design philosophy is:

`Project coarsely` $\rightarrow$ `Verify precisely` $\rightarrow$ `Publish confidently`

The estimator's role is to provide a reproducible, index-addressable candidate space. The verification layer (Section 3) and primality confirmation ensure that only valid safe primes are published.

3 Structural Verification via Lucas' Theorem

Each candidate produced by the estimator is evaluated using a structural verification layer before primality confirmation. This layer produces a numeric *collapse score* in the range $[0, 1]$ that serves as a deterministic primality proxy.

3.1 Mathematical Basis

The scoring mechanism is based on a well-known consequence of Lucas' theorem [7]: for any prime p and integer k with $1 \leq k \leq T$ (where T is a fixed window size, typically $T = 128$), the binomial coefficient $\binom{p}{k}$ is divisible by p. Equivalently, $\binom{p}{k} \equiv 0 \pmod{p}$ for all $1 \leq k \leq p-1$.

For composite numbers, this property generally fails for at least some values of k within the window. The collapse score measures the fraction of the window over which the divisibility property holds:

$$S(n, T) = \frac{1}{T} \sum_{k=1}^{T} \mathbf{1}\left[\binom{n}{k} \equiv 0 \pmod{n} \right]$$

For any prime $p > T$, all binomial coefficients $\binom{p}{k}$ for $1 \leq k \leq T$ are divisible by p, so $S(p, T) = 1.0$. This is a direct consequence of Lucas' theorem and is not novel mathematics.

The value of the score is not as a primality test—standard deterministic tests are more efficient for that purpose—but as a *structural verification trace*: a reproducible numeric value that is permanently attached to each published prime, providing an auditable record of structural evaluation.

3.2 Role in the Pipeline

The collapse score serves several purposes within EchoPrime:

- **Auditable trace.** Each published prime carries a numeric score that any party can independently recompute and verify.
- **Composite filtering.** Composite candidates generally receive scores below 1.0, enabling efficient pre-filtering before expensive primality confirmation.
- **On-chain metadata.** The score is published alongside the prime in the oracle contract, providing queryable structural metadata for downstream consumers.

After scoring, candidates undergo standard deterministic primality testing [8, 9] as a final confirmation. Both p and $q = (p-1)/2$ must pass primality testing to be published. The collapse score and primality verdict together form the complete verification trace.

4 Million-Scale Validation

To assess the estimator and verification pipeline at scale, we tested 1,000,000 projected safe prime candidates.

4.1 Experimental Setup

Each candidate was generated by the EchoPrime estimator from sequential indices. For each candidate pair (p, q) where $p = 2q + 1$:

1. The collapse score was computed over a window $T = 128$.
2. Candidates were accepted if $S(p, 128) \geq 0.95$ and $S(q, 128) \geq 0.95$.
3. Accepted candidates were confirmed via deterministic primality testing.

4.2 Results

Of the 1,000,000 projected candidates, 76,144 were valid safe primes—a hit rate of approximately 7.6%.

Verified: 76,144 / 76,144 candidates passing threshold (100.00% confirmed prime)

All 76,144 candidates that passed the collapse score threshold were confirmed as valid safe primes by deterministic primality testing. No false positives were recorded: every candidate with $S(p, 128) \geq 0.95$ and $S(q, 128) \geq 0.95$ was indeed a safe prime.

4.3 Estimator Efficiency

The hit rate reflects the estimator's ability to target regions of the number line where safe primes cluster. At the 1,000,000-candidate scale, the observed rate was approximately 7.6%; independent verification at 10,000 candidates yielded 9.74%, consistent with scale-dependent density variation. For context, the asymptotic density of safe primes among odd numbers at comparable scales is approximately 0.23%, meaning the estimator is roughly 33–42× more efficient than uniform random sampling at the tested scale (∼30-bit candidates).

Note: These benchmarks were conducted at approximately 30-bit scale. Efficiency at larger bit sizes (e.g., 1024-bit) has not been empirically validated and would depend on recalibrating the estimator constant α for larger ranges. We do not claim efficiency gains at scales beyond those tested.

4.4 Verifier Accuracy

The 100% confirmation rate across 76,144 candidates demonstrates that the collapse score threshold of 0.95 is an effective pre-filter: it admits all true safe primes in the candidate space while excluding composites. This is expected from the Lucas' theorem basis—primes above T will always score 1.0—but the empirical validation confirms consistent behavior across a large sample.

5 Oracle Contract Architecture

The core of the EchoPrime system is a smart contract deployed on an Ethereum-compatible blockchain. This contract serves as a public registry for verified safe prime submissions.

5.1 Storage and Submission

The contract maintains a mapping:

$$\texttt{records[index]} \rightarrow (p, q, \texttt{scoreP}, \texttt{scoreQ}, \texttt{verified})$$

Each entry corresponds to a single safe prime submission, indexed by the deterministic projection index. The stored tuple includes:

- p: the safe prime
- $q = \frac{p-1}{2}$: the corresponding Sophie Germain prime
- `scoreP`, `scoreQ`: collapse scores for p and q
- `verified`: boolean indicating confirmed primality of both p and q

Submissions are recorded via:

```
function submitVerification(
    uint256 index,
    uint256 p,
    uint256 scoreP,
    uint256 scoreQ,
    bool verified
) external;
```

Any party—ceremony coordinator, oracle bot, or protocol—can publish a safe prime and its trace. Upon submission, the contract stores the record and emits a `PrimeVerified` event.

5.2 Event Trace

Every submission emits a standardized event:

```
event PrimeVerified(
    uint256 index,
    uint256 p,
    uint256 scoreP,
    uint256 scoreQ,
    bool verified
);
```

These logs are permanently recorded on-chain and can be consumed by off-chain indexers, ZK rollup coordinators, or ceremony orchestration systems. Observers can monitor for newly published primes, filter by score threshold, or confirm whether a particular index has been verified.

5.3 Query Interface

The registry supports public reads:

```
function getPrime(uint256 index) returns PrimeRecord
```

This enables on-chain and off-chain systems to retrieve verified primes and their associated metadata for downstream use in cryptographic protocols.

6 Comparison to Existing Methods

6.1 Conventional Safe Prime Generation

Standard safe prime generation follows a probabilistic process:

1. Sample a random odd integer q
2. Test if q is prime (typically Miller–Rabin [8, 9])
3. Compute $p = 2q + 1$

4. Test if p is prime

5. Repeat until success

This process depends on entropy sources, yields no structural trace, and produces outputs that cannot be deterministically reconstructed by downstream verifiers.

6.2 Comparison Table

Feature	**Conventional Tools**	**EchoPrime**
Generation method	RNG + trial/sieve	Deterministic projection from index
Primality testing	Probabilistic (Miller–Rabin)	Structural scoring + deterministic confirmation
Reproducibility	Non-deterministic	Fully deterministic from index
Audit trail	None	On-chain trace with scores and index
Setup transparency	Opaque	Publicly verifiable registry
Index → prime mapping	Not supported	Direct, deterministic

6.3 What EchoPrime Does Not Replace

EchoPrime is not a replacement for standard primality testing algorithms or general-purpose cryptographic libraries. Tools like OpenSSL, GMP, and libsodium provide efficient, well-audited implementations of prime generation and testing. EchoPrime's contribution is the combination of deterministic index-based projection with on-chain auditability—creating a verifiable provenance layer on top of standard cryptographic operations.

7 Use Cases

7.1 Trusted Setup Ceremonies

Zero-knowledge proving systems [1] (zkSNARKs, PLONK variants) require trusted setup ceremonies where cryptographic parameters must be generated transparently. EchoPrime allows ceremony coordinators to derive parameters from publicly auditable, collapse-verified primes:

$$\texttt{curveSeed} \leftarrow \texttt{hash}(p \,\|\, \texttt{scoreP} \,\|\, \texttt{index})$$

This eliminates reliance on opaque entropy sources during setup and enables any participant to verify the derivation.

7.2 MPC Group Parameters

Multiparty computation protocols [2] require agreement on group parameters, typically safe primes for the group modulus. EchoPrime provides pre-verified, publicly registered safe primes that all parties can independently confirm:

$$g \in \mathbb{Z}_p^*, \quad p \in \texttt{EchoPrime registry}$$

This avoids disputes over parameter sourcing in threshold cryptography, distributed key generation, and secure multiparty signing.

7.3 Diffie–Hellman and RSA Parameters

Safe primes are directly used in Diffie–Hellman key exchange [3] (as the group modulus) and in RSA key generation [4] (as prime factors of $N = pq$). EchoPrime provides a transparent source of such primes with verifiable provenance, suitable for applications where parameter auditability is required.

7.4 ZK Rollup Infrastructure

Layer-2 rollup systems require transparent, verifiable cryptographic parameters for curve selection, proof system initialization, and trusted setup registries. EchoPrime's on-chain registry allows rollup protocols to reference publicly recorded safe primes with full trace metadata, reducing trust assumptions in parameter sourcing.

7.5 Deterministic Key Derivation

EchoPrime's index-to-prime mapping enables deterministic key derivation schemes where cryptographic material is tied to a publicly auditable symbolic index rather than opaque entropy. This supports reproducible wallet generation, multisig recovery, and identity key derivation with verifiable provenance.

8 Cross-Chain Extensions

While EchoPrime is currently deployed on Ethereum-compatible chains, its deterministic architecture is blockchain-agnostic.

8.1 Bitcoin Integration

Bitcoin's limited scripting does not support complex on-chain verification, but EchoPrime traces can be anchored to Bitcoin via several mechanisms:

- **OP_RETURN commitments.** A SHA-256 hash of the prime trace $(p, q, \texttt{score}_p, \texttt{score}_q, \texttt{index})$ can be published in an `OP_RETURN` output, creating a timestamped proof of existence on Bitcoin.

- **Sidechain deployment.** Bitcoin-compatible sidechains such as Rootstock (RSK) and Stacks support smart contracts and could host a full EchoPrime oracle instance.
- **Cross-chain relay.** Oracle protocols such as Chainlink or LayerZero can relay EchoPrime traces from Ethereum to Bitcoin-aware systems.

8.2 Applications in Bitcoin Infrastructure

- **Threshold signatures (MuSig2, FROST):** EchoPrime primes can serve as verifiable group parameters for Taproot-based threshold schemes.
- **Cross-chain custody:** MPC key shares across Bitcoin and EVM chains can coordinate around EchoPrime-verified safe primes.
- **Audit-trail anchors:** OP_RETURN commitments provide decentralized proofs-of-trace for compliance and recovery workflows.

9 Future Work

Planned development focuses on extending the system's reach and robustness:

- **Large-scale empirical validation.** Extend benchmarks to 512-bit and 1024-bit candidates with recalibrated estimator constants, and publish results.
- **Multi-chain deployment.** Deploy oracle instances on additional EVM chains and Layer-2 rollups to broaden accessibility.
- **Public API and SaaS access.** Provide a hosted API for prime generation, verification, and registry queries, enabling integration without direct smart contract interaction.
- **Enterprise features.** Support batch submissions, custom score thresholds, and private oracle instances for organizations with specific compliance or parameter requirements.
- **Extended scoring operators.** Investigate alternative structural verification metrics beyond the Lucas' theorem-based collapse score, including tunable window sizes and weighted scoring schemes.
- **Formal security analysis.** Commission independent review of the estimator distribution properties and the verification pipeline's false-positive guarantees at cryptographic scales.

10 Conclusion

EchoPrime provides a deterministic, auditable pipeline for safe prime generation and on-chain publication. Its estimator maps natural number indices to candidate safe primes; its verification layer, based on Lucas' theorem, attaches a reproducible structural score to each candidate; and its oracle contract permanently records verified primes with full trace metadata on an Ethereum-compatible blockchain.

The system does not claim to replace established cryptographic libraries or primality testing algorithms. Its contribution is the combination of deterministic index-based projection with on-chain auditability and structural verification traces. This creates a publicly inspectable, reproducible record of cryptographic parameter provenance—a property not provided by conventional generation methods.

Empirical validation across 1,000,000 candidates confirms a 7.6%–9.7% safe prime hit rate (approximately 33–42× more efficient than random sampling at the tested 30-bit scale) with 100% verification accuracy across all accepted candidates. The on-chain oracle architecture enables composable integration with ZK rollups, MPC protocols, trusted setup ceremonies, and deterministic key derivation systems.

EchoPrime is available as a public oracle on Ethereum-compatible chains. Documentation, contract addresses, and API access are provided at `www.echoprime.xyz`.

References

[1] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von Neumann architecture," in *Proceedings of the 23rd USENIX Security Symposium*, pp. 781–796, 2014.

[2] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," *Journal of Cryptology*, vol. 20, no. 1, pp. 51–83, 2007.

[3] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[4] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[5] G. H. Hardy and J. E. Littlewood, "Some problems of 'Partitio Numerorum'; III: On the expression of a number as a sum of primes," *Acta Mathematica*, vol. 44, pp. 1–70, 1923.

[6] P. T. Bateman and R. A. Horn, "A heuristic asymptotic formula concerning the distribution of prime numbers," *Mathematics of Computation*, vol. 16, no. 79, pp. 363–367, 1962.

[7] É. Lucas, "Théorie des fonctions numériques simplement périodiques," *American Journal of Mathematics*, vol. 1, no. 2, pp. 184–196, 1878.

[8] G. L. Miller, "Riemann's hypothesis and tests for primality," *Journal of Computer and System Sciences*, vol. 13, no. 3, pp. 300–317, 1976.

[9] M. O. Rabin, "Probabilistic algorithm for testing primality," *Journal of Number Theory*, vol. 12, no. 1, pp. 128–138, 1980.

[10] A. Granville, "Harald Cramér and the distribution of prime numbers," *Scandinavian Actuarial Journal*, vol. 1995, no. 1, pp. 12–28, 1995.