The Architecture of Trust: An Exhaustive Technical Analysis of the drand Distributed Randomness Beacon and the League of Entropy

1. Introduction: The Deterministic Paradox and the Quest for Entropy

In the rigid logic of computational systems, randomness is a paradox. Computers are deterministic machines designed to execute instructions with precise repeatability; asking such a machine to behave unpredictably is inherently contradictory. Yet, unpredictability—specifically, verifiable, unbiasable, and high-entropy randomness—is the lifeblood of the modern digital economy. It is the fundamental resource required to secure cryptographic keys, arbitrate leader elections in decentralized networks, ensure fairness in digital lotteries, and parameterize complex audit processes.

For decades, the industry relied on flawed surrogates. Systems utilized pseudo-random number generators (PRNGs) seeded by local, often low-entropy sources like thermal noise or interrupt timings, creating vectors for prediction attacks. Alternatively, they relied on centralized "beacons" such as the service provided by the National Institute of Standards and Technology (NIST), which, while cryptographically competent, introduced a single point of failure and required users to inherently trust a government agency not to manipulate or pre-release the output. The "Oracle Problem" in blockchain—how to bring off-chain data (like randomness) onto the chain without trusting a central intermediary—remained a critical bottleneck.

The **drand** (distributed randomness) project and the associated **League of Entropy** (LoE) consortium represent the industrial-grade solution to this dilemma. By shifting the generation of randomness from a single hardware device to a distributed network of independent nodes, drand creates a "Randomness-as-a-Service" (RaaS) utility. It leverages **Threshold Cryptography** and **Multi-Party Computation (MPC)** to ensure that no single entity, nor even a conspiring minority of entities, can predict or bias the output. This report provides an exhaustive, expert-level analysis of the drand protocol, its cryptographic underpinnings, the operational structure of the League of Entropy, and the broader implications for the Web3 and cybersecurity landscapes.

2. Theoretical Foundations: The Mathematics of Distributed Trust

To understand the robustness of drand, one must first dissect the cryptographic primitives that render it secure. The system does not merely "mix" random numbers; it constructs a mathematical object—a threshold signature—that possesses unique properties of determinism

and unforgeability.

2.1 Threshold Cryptography and the t-of-n Trust Model

The core security architecture of drand is built upon **Threshold Cryptography**. In a standard public-key cryptosystem, a single private key sk corresponds to a public key pk. Possession of sk allows an entity to sign messages or decrypt data. If sk is compromised, the entire system collapses.
Drand employs a t-of-n threshold model. Here, the "private key" is not a single string of data held in one location. Instead, it is a mathematical abstraction that exists only potentially. The key is effectively split into n shares (s_1, s_2,..., s_n) and distributed among n independent nodes. To perform a cryptographic operation—specifically, generating a BLS signature that serves as the randomness beacon—a subset of at least t nodes (the threshold) must contribute their partial signatures.

2.1.1 The Security Guarantee: f < n

The system assumes a threat model where at most f nodes are malicious or compromised, where f < t. The threshold is typically set such that t = f + 1 or higher (often ensuring an honest majority).

- **Secrecy:** Any subset of nodes smaller than t learns absolutely nothing about the collective private key. Even if an attacker compromises t-1 nodes, the randomness remains mathematically unpredictable.
- **Liveness:** As long as t honest and functional nodes are available, the network can continue to produce beacons. This provides resilience against Denial-of-Service (DoS) attacks or random node failures.

2.2 Pairing-Based Cryptography (PBC)

Drand utilizes **Pairing-Based Cryptography**, a specialized branch of elliptic curve cryptography that enables operations impossible in standard ECC. The system relies on bilinear groups (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T) of prime order p.
The defining feature is the **bilinear pairing** map e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T. This map must satisfy three critical properties:

1. **Bilinearity:** For all P \in \mathbb{G}_1, Q \in \mathbb{G}_2 and all a, b \in \mathbb{Z}, the equation e(aP, bQ) = e(P, Q)^{ab} holds. This allows the exponents (the secret keys) to be moved effortlessly between the groups during calculation, a property vital for verifying threshold signatures.
2. **Non-degeneracy:** The pairing is non-trivial; e(P, Q) \neq 1 for generators P, Q.
3. **Computability:** There exists an efficient algorithm to compute e(P, Q).

2.2.1 The BLS12-381 Curve

Drand exclusively uses the **BLS12-381** curve. This pairing-friendly curve was selected for its balance of security (approximately 128-bit security level) and performance. It is constructed to allow for efficient implementation of the pairing function while resisting attacks like the Discrete Logarithm Problem (DLP) in the extension field.

- **\mathbb{G}_1 vs. \mathbb{G}_2:** In pairing-based schemes, elements in \mathbb{G}_1

are typically smaller and faster to compute than those in \mathbb{G}_2. Drand's configurable schemes (discussed in Section 4) exploit this asymmetry to optimize for either signature size or verification speed.

2.3 Boneh-Lynn-Shacham (BLS) Signatures

The specific application of PBC in drand is the **Boneh-Lynn-Shacham (BLS) signature scheme**. BLS signatures are uniquely suited for distributed randomness for two reasons: **uniqueness** and **aggregatability**.

1. **Determinism (Uniqueness):** For a given key pair and a given message, there is exactly one valid BLS signature. Unlike ECDSA, which requires a random nonce (introducing potential bias or side-channel leakage), BLS signatures are deterministic. This property is the cornerstone of drand's "Unbiasability". No matter which t nodes contribute to the threshold signature, the resulting aggregate signature is always identical.
2. **Aggregation:** BLS allows for the aggregation of multiple signatures into a single signature. In a threshold context, partial signatures \sigma_i generated by individual nodes can be Lagrange-interpolated to reconstruct the full group signature \sigma.

3. Technical Specification of the drand Beacon

The drand protocol is implemented as a system daemon (written in Go) that orchestrates the complex interaction of networking, storage, and cryptography.

3.1 Anatomy of a Beacon

A drand beacon is not merely a random number; it is a structured cryptographic artifact generated at a specific point in time, known as a **Round**. The beacon structure generally contains:

- **Round:** The monotonically increasing index of the randomness generation (uint64).
- **Signature:** The BLS signature computed by the network for this round. This raw byte sequence *is* the source of randomness. Because it is the output of a secure signature scheme on a high-entropy curve, it is uniformly distributed and indistinguishable from random noise to anyone without the key.
- **PreviousSignature:** (In Chained Mode) The signature of the beacon at Round R-1. This cryptographically binds the chain history.
- **Randomness:** A hash (usually SHA-256) of the Signature. This provides a fixed-length, uniform output easy for applications to consume.

3.2 Network Phases

The lifecycle of a drand network is strictly segmented into phases to ensure security setup and continuous operation.

3.2.1 Setup Phase: Distributed Key Generation (DKG)

Before the first beacon is emitted, the network must establish its collective identity. Drand utilizes a **Distributed Key Generation (DKG)** protocol, specifically **Pedersen's DKG with**

Feldman's Verifiable Secret Sharing (VSS). This phase eliminates the need for a "trusted dealer" who creates keys and hands them out (a centralized failure mode).

1. **Contribution:** Each node i generates a random polynomial f_i(x) of degree t-1. The constant term f_i(0) is the node's secret contribution to the master key.
2. **Commitment:** To allow verification without revealing secrets, nodes broadcast commitments to the coefficients of their polynomials.
3. **Distribution:** Each node sends encrypted shares to every other node. Node j receives f_i(j) from node i.
4. **Verification and Complaint:** Nodes verify the shares against the public commitments. If a share is invalid, the node issues a complaint. The protocol includes mechanisms to disqualify malicious dealers.
5. **Finalization:** Each node computes its final long-term secret share s_i by summing the valid shares received from all other nodes: s_i = \sum_j f_j(i). The collective public key is the sum of all public constant term commitments.

3.2.2 Generation Phase (The Beacon Loop)

Once Setup is complete, the network enters the Generation phase, operating in an infinite loop defined by the **Period** (the frequency of beacons, e.g., 3 seconds or 30 seconds).

1. **Proposal:** A node (or all nodes, depending on implementation) initiates the round.
2. **Partial Signing:** Each node i constructs the message M for the current round R.
    - *Chained Mode:* M = H(R \parallel \sigma_{R-1}).
    - *Unchained Mode:* M = H(R).
    - Using its share s_i, the node computes a partial signature \sigma_i.
3. **Broadcast:** The partial signature is broadcast to the network using the Gossipsub protocol.
4. **Reconstruction:** Any node that collects t valid partial signatures can reconstruct the full signature \sigma using Lagrange interpolation.
5. **Dissemination:** The full beacon is published via public HTTP relays, Gossipsub topics, and other distribution channels.

3.3 Scheme Configurations

Drand is highly configurable, supporting different "Schemes" that dictate the cryptographic parameters. The choice of scheme affects beacon size, verification speed, and feature compatibility.

3.3.1 pedersen-bls-chained (Legacy Default)

This was the original standard for the League of Entropy.

- **Chaining:** Enabled. The beacon at round R is dependent on R-1. This creates a "blockchain-like" immutability; an attacker cannot fork the randomness history without rewriting all subsequent rounds.
- **Groups:** Signatures are on \mathbb{G}_2 (96 bytes), Public Keys on \mathbb{G}_1 (48 bytes).
- **Drawback:** Verification is stateful. To verify round N, a client theoretically needs the chain of signatures leading back to a trusted checkpoint. This is computationally expensive for light clients.

3.3.2 bls-unchai[span_21](start_span)[span_21](end_span)ned-g1-rfc9380 (Modern Standard / Quicknet)

This scheme represents the current state-of-the-art, used in the **Quicknet** network.

- **Chaining:** Disabled (Unchained). The message signed is simply the round number.
- **Groups:** Swapped. Signatures are on \mathbb{G}_1 (48 bytes), Public Keys on \mathbb{G}_2 (96 bytes).
- **Benefits:**
    1. **Size:** Beacons are 50% smaller (48 bytes vs 96 bytes), drastically reducing storage costs on consuming blockchains like Filecoin.
    2. **Stateless Verification:** A client can verify any round R instantly using only the group public key, without knowledge of previous rounds.
    3. **Timelock Compatibility:** Because the message (Round R) is known in advance, this scheme enables Timelock Encryption (see Section 6).
- **Compliance:** This scheme strictly adheres to **RFC 9380** for Hashing to Curves, ensuring interoperability and security vetting.

4. Security Analysis: Unbiasability and Attack Mitigation

The primary value proposition of drand is its resistance to manipulation. The user query specifically asks how the system prevents "predicting or biasing" a draw.

4.1 Unpredictability: The t-of-n Barrier

Predicting the randomness is equivalent to forging a signature on the message M without the private key. In drand, the private key does not exist in any single memory space.

- **The Attack Vector:** To predict the outcome of Round R+1 before time T_{R+1}, an attacker would need to compromise t distinct nodes, extract their secret shares, and combine them.
- **Mitigation:** The geographic and organizational diversity of the League of Entropy (discussed in Section 5) makes this practically impossible. Compromising a server at Cloudflare, a research workstation at EPFL, and a node at the University of Chile simultaneously—without detection—is a complexity class far beyond standard attacks.

4.2 Unbiasability: Defeating the Rushing Adversary

In many distributed randomness protocols (e.g., "commit-and-reveal" schemes), a **Rushing Attacker** poses a severe threat. A rushing attacker is the last participant to speak. They observe the contributions of all other nodes, calculate the potential result, and then decide whether to reveal their own contribution or withhold it to influence the outcome (or abort the protocol).

Drand neutralizes the Rushing Attack through the properties of BLS signatures:

1. **Uniqueness:** For a given group public key and a message (Round R), there is only *one* mathematically valid signature.
2. **No Choice:** A malicious node M has no freedom to "choose" a contribution that shifts the

randomness. M's share is mathematically fixed by the DKG setup. M can compute \sigma_M, but they cannot compute a \sigma'_M that is valid but different.

3. **Withholding is Futile:** If the attacker sees the partial signatures of honest nodes and realizes the resulting beacon will be unfavorable (e.g., they lose the lottery), they can refuse to broadcast their share. However, because the network is t-of-n redundant (where n > t), the honest nodes will simply reconstruct the *same* unique signature without the attacker. The result is identical whether the attacker participates or not.

Conclusion: The attacker cannot change the value (bias) and cannot stop the value from being produced (liveness), provided t honest nodes exist.

4.3 Resilience to Replay and Forking

- **Replay Attacks:** Messages in drand are signed over the Round Number (and optionally the Chain ID). A signature from Round 10 cannot be replayed as a valid beacon for Round 11, as the message hash would differ.
- **Chain ID Separation:** A single node often runs multiple networks (e.g., Mainnet and Testnet). These are cryptographically separated by **Chain Hashes** and distinct **Beacon IDs**. A message intended for the testnet cannot influence the mainnet, even if processed by the same binary.

5. The League of Entropy: Industrial-Scale Reliability

While the drand protocol provides theoretical security, the **League of Entropy (LoE)** provides the physical and operational security required for a production-grade utility. Established in 2019, the LoE is a consortium of independent organizations that collaborate to operate the drand mainnet.

The League functions as a "Public Good" infrastructure. No single member owns the network; trust is distributed among the participants.

5.1 Member Organizations: A Profile of Diversity

The League's strength lies in the heterogeneity of its members. They span different continents, legal jurisdictions, tech stacks, and industries (academic, corporate, non-profit). This diversity ensures that no single subpoena, hardware bug, or regional internet outage can compromise the entire network.

Confirmed List of Participating Organizations: Based on the latest available data, the following organizations operate nodes in the League of Entropy :

1. **Cloudflare:** A global web performance and security company. Their participation proves the scalability of the network.
2. **EPFL (École Polytechnique Fédérale de Lausanne):** The Swiss university where the DEDIS lab (creators of drand) is based.
3. **Protocol Labs:** The research and development lab behind IPFS and Filecoin, a primary consumer of the beacon.
4. **Kudelski Security:** A specialized cybersecurity division of the Kudelski Group, bringing auditing and high-security operations expertise.
5. **University of Chile (UChile):** Participating via their "Random UChile" project.
6. **ChainSafe:** A leading blockchain R&D firm, demonstrating deep Web3 integration.

7. **cLabs:** The team developing the Celo blockchain.
8. **Ethereum Foundation:** The non-profit organization supporting the Ethereum ecosystem.
9. **Emerald Onion:** A 501(c)(3) non-profit transit internet service provider and privacy advocate.
10. **IC3 (The Initiative for Cryptocurrencies and Contracts):** An academic collaboration involving Cornell, UC Berkeley, and others.
11. **Tierion:** A company focusing on blockchain data verification (Chainpoint).
12. **UCL (University College London):** Another academic node enhancing the research credibility of the network.
13. **C4DT (Center for Digital Trust):** Located at EPFL, focusing on trust technologies.
14. **PTisp:** A Portuguese hosting provider.
15. **QRL Foundation:** The Quantum Resistant Ledger project.
16. **StorSwift:** A decentralized cloud storage provider.
17. **IPFSForce:** A major Filecoin storage provider.
18. **KEN Labs:** A distributed systems research lab.
19. **Automata Network:** A middleware provider focusing on privacy and fair computation.

5.2 Entropy Sources: Harnessing Physical Chaos

A unique feature of the League is how its members seed the randomness. Rather than relying solely on CPU interrupts, members inject entropy from the physical world, creating a "mixed" entropy pool that is impossible to model.

- **LavaRand (Cloudflare):** Perhaps the most famous source. Cloudflare aims a camera at a wall of lava lamps in their San Francisco lobby. The unpredictable fluid dynamics, combined with lighting changes and passersby, generate high-quality entropy.
- **Seismic Girl (University of Chile):** This source draws entropy from a network of seismic sensors monitoring earthquake activity in Chile. It is literally "earth-shaking" randomness.
- **ChaChaRand (Kudelski):** Utilizes a cryptographic RNG based on the ChaCha20 stream cipher, providing a high-speed, cryptographically secure baseline.
- **URand (EPFL):** Relies on the local randomness generator (/dev/urandom) of the server, mixing in system noise.

The DKG protocol ensures that this entropy is uniformly mixed. Even if one source is completely predictable (e.g., the lava lamps freeze), the resulting key remains secure as long as *one* source provided true entropy.

6. Network Evolution, Outages, and Recovery

The League of Entropy operates multiple networks, iterating on technology while maintaining uptime.

6.1 The Transition: Fastnet to Quicknet

The League initially operated a "Fastnet" designed for low-latency applications. However, as cryptographic standards matured, specifically with the publication of **RFC 9380**, Fastnet was found to be using non-compliant Hash-to-Curve parameters.

- **The Sunset:** In late 2024, Fastnet was deprecated. The League executed "scream tests"—brief, intentional outages—to force relying applications to migrate.

- **The Replacement (Quicknet):** Quicknet was launched as the new standard. It operates at a **3-second frequency** (unprecedented for distributed beacons) and uses the unchained, RFC-compliant scheme. It runs on the same physical nodes as the previous networks but as a separate logical process.

6.2 Incident Analysis: The February 2024 Testnet Outage

While the Mainnet has maintained 100% uptime in recent records , the Testnet serves as a proving ground where failures provide valuable lessons. In February 2024, a Testnet ceremony stalled.

- **Root Cause 1:** A node operator (KEN Labs) inadvertently overwrote their cryptographic keypair while attempting to update their node, effectively "forgetting" their identity.
- **Root Cause 2:** Another node (QRL) experienced a configuration drift where it fell back to non-TLS connections, while the rest of the network enforced TLS. This partitioned the network.
- **Resolution:** The incident forced a manual intervention (DKG "nuke" and restart) but resulted in software patches that now prevent accidental key overwrites. This proves the value of the "Testnet first" strategy, ensuring such operational errors do not impact the Mainnet.

7. Advanced Capabilities: Timelock Encryption

Drand's transition to Unchained Randomness enabled a capability that was previously theoretical: **Timelock Encryption** (or Identity-Based Encryption with Time). This allows users to encrypt data such that it can *only* be decrypted after a specific future time.

7.1 The Mechanism: Identity-Based Encryption (IBE)

Drand implements this using the Boneh-Franklin IBE scheme (or similar pairing-based variants).

- **The Identity:** In IBE, a public key can be any string. Drand uses the **Round Number** as the identity.
- **The Private Key:** The private key corresponding to that identity is the **Beacon Signature** for that round.
- **The Flow:**
    1. **Encrypt:** Alice wants to encrypt a file for next Friday. She calculates the round number R for that time. She encrypts the file using the League's Group Public Key and the ID "R".
    2. **Wait:** At this point, the ciphertext is locked. The key to unlock it (Signature R) does not exist yet.
    3. **Decrypt:** When Friday arrives, the League generates Signature R. Any observer can grab this signature, input it into the decryption algorithm, and unlock the file.

7.2 Integration with age and tlock

Because IBE is computationally intensive and restricted in message size, drand provides tools like tlock that utilize **Hybrid Encryption**.

- The actual file is encrypted using a symmetric key (via the age tool standards).

- The symmetric key is encrypted via drand's IBE scheme.
- This allows efficient encryption of large files "towards the future" using standard command-line interfaces.

8. Industrial Integration and Use Cases

Drand is not an academic curiosity; it is the heartbeat of major decentralized infrastructures.

8.1 Filecoin: Securing Storage Consensus

Filecoin, the decentralized storage network, is the largest consumer of drand.

- **Leader Election:** Filecoin uses **Secret Leader Election** based on Expected Consensus. To determine if a miner is eligible to produce a block in a given epoch, the miner must sample randomness.
- **Why drand?** If miners generated their own randomness (e.g., via block hashes), they could "grind" (retry repeatedly) to find a value that lets them win. By using drand, the randomness is external, fixed, and unbiasable. A miner checks the drand value; if they didn't win, they cannot retry. This guarantees fairness proportional to storage power.

8.2 The Cosmos Ecosystem: Nois Network

In the Cosmos blockchain ecosystem, the **Nois Network** acts as a randomness bridge. Verifying BLS signatures on-chain is computationally expensive (high gas costs).

- **The Solution:** Nois fetches the drand beacon, verifies it *once* on its own chain, and then transmits the verified random seed to other Cosmos chains (like Juno or Stargaze) via the Inter-Blockchain Communication (IBC) protocol. This provides "Randomness Oracle" services to hundreds of gaming and lottery dApps efficiently.

8.3 Gaming and MEV Protection

- **Fairness:** Web3 games use drand to determine loot drop rates and shuffle decks, providing cryptographic proof to players that the admin did not rig the game.
- **MEV (Maximal Extractable Value) Protection:** Timelock Encryption is being explored to encrypt transactions in a "mempool." By encrypting a transaction until it is included in a block, the transaction content is hidden from miners, preventing them from "front-running" or sandwiching the trade. The transaction is only revealed (decrypted) once the ordering is finalized.

9. Conclusion

The drand Distributed Randomness Beacon represents a paradigm shift in how digital systems consume entropy. By effectively "decentralizing trust," it resolves one of the oldest problems in computer science: how to generate unpredictability in a deterministic environment without relying on a central authority.

Through the League of Entropy, drand has achieved industrial-scale reliability. The collaboration of competitors (like cloud providers and blockchain networks) in running the infrastructure proves that security can be a non-zero-sum game. With the adoption of RFC-compliant

schemes like Quicknet and the unlocking of novel utilities like Timelock Encryption, drand has established itself as a foundational layer of the Web3 stack—an immutable, unbiasable clock that beats every three seconds, securing billions of dollars in value across the digital ecosystem.

Table 1: Drand Network Specifications (Mainnet)

Feature	**Default Network** (Legacy)	**Quicknet** (Current Standard)
Period (Frequency)	30 Seconds	**3 Seconds**
Mode	Chained	**Unchained**
Scheme ID	pedersen-bls-chained	bls-unchained-g1-rfc9380
Signature Group	\mathbb{G}_2 (96 bytes)	\mathbb{G}_1 (48 bytes)
Public Key Group	\mathbb{G}_1 (48 bytes)	\mathbb{G}_2 (96 bytes)
Timelock Compatible	No	**Yes**
Primary Use Case	Archival / Historical Verification	High-Frequency / Smart Contracts

Table 2: Key Cryptographic Primitives

Component	Specification	Purpose
Curve	**BLS12-381**	Pairing-friendly curve offering ~128-bit security.
Signature Scheme	**Threshold BLS**	Allows signature aggregation and t-of-n generation.
DKG Protocol	**Pedersen + Feldman VSS**	Generates distributed keys without a trusted dealer.
Hash-to-Curve	**RFC 9380**	Standardized mapping of bytes to curve points.
Timelock Scheme	**Boneh-Franklin IBE**	Uses Identity-Based Encryption for time-release data.

Quellenangaben

1. League of entropy - Drand, https://www.drand.love/loe 2. Public Randomness and Randomness Beacons - a16z crypto, https://a16zcrypto.com/posts/article/public-randomness-and-randomness-beacons/ 3. drand Explained, https://docs.drand.love/about/ 4. Inside the Entropy - The Cloudflare Blog, https://blog.cloudflare.com/inside-the-entropy/ 5. Cryptography | drand, https://docs.drand.love/docs/cryptography/ 6. drand/drand: A Distributed Randomness Beacon Daemon - Go implementation - GitHub, https://github.com/drand/drand 7. Protocol Specification | drand, https://docs.drand.love/docs/specification/ 8. Threshold Cryptography and the Decipher Network: Many Hands, One Key - Medium, https://medium.com/@naoddemiseasefa/threshold-cryptography-and-the-decipher-network-many-hands-one-key-575b45a60c03 9. Randomness Generation · Cloudflare Randomness Beacon docs, https://developers.cloudflare.com/randomness-beacon/cryptographic-background/randomness-generation/ 10. drand - Distributed Randomness - Filecoin Spec, https://spec.filecoin.io/libraries/drand/ 11. The random number generator for the IBC world | by Simon Warta - Medium, https://medium.com/@simonwarta/the-random-number-generator-for-the-ibc-world-c0eea409dd

27 12. `quicknet` is live on the League of Entropy mainnet - drand, https://docs.drand.love/blog/2023/10/16/quicknet-is-live/ 13. Timelock Encryption - drand, https://docs.drand.love/docs/timelock-encryption/ 14. `fastnet` sunsetting dates are set - drand, https://docs.drand.love/blog/fastnet-to-be-sunset/ 15. Security Model | drand, https://docs.drand.love/docs/security-model/ 16. OptRand: Optimistically Responsive Reconfigurable Distributed Randomness - NDSS Symposium, https://www.ndss-symposium.org/wp-content/uploads/2023-832-paper.pdf 17. drand HTTP API, https://docs.drand.love/dev-guide/API%20Documentation%20v1/drand-http-api/ 18. The League of Entropy Launches Production Drand Network, Providing the First Publicly Verifiable Distributed Randomness Beacon - PR Newswire, https://www.prnewswire.com/news-releases/the-league-of-entropy-launches-production-drand-network-providing-the-first-publicly-verifiable-distributed-randomness-beacon-301109096.html 19. Distributed Randomness Beacon - Cloudflare, https://www.cloudflare.com/leagueofentropy/ 20. Tierion Joins the League of Entropy — Replaces NIST Beacon with Drand in Chainpoint, https://blog.tierion.com/tierion-joins-the-league-of-entropy-replaces-nist-beacon-with-drand-in/ 21. League of Entropy: Not All Heroes Wear Capes - The Cloudflare Blog, https://blog.cloudflare.com/league-of-entropy/ 22. Harnessing chaos in Cloudflare offices, https://blog.cloudflare.com/harnessing-office-chaos/ 23. `fastnet` is being sunset, long live `quicknet` | drand, https://docs.drand.love/blog/2023/07/03/fastnet-sunset-quicknet-new 24. drand Status, https://status.drand.love/ 25. Uptime History - drand Status, https://status.drand.love/uptime 26. One post tagged with "Post Mortem" - drand, https://docs.drand.love/blog/tags/post-mortem 27. Post-mortem following testnet outage on 21st of February - drand, https://docs.drand.love/blog/2024/02/23/testnet-ceremony-post-mortem/ 28. dee: Rust cli for drand - Crates.io, https://crates.io/crates/dee 29. Timelock Encryption: An Overview and Retrospective, https://csrc.nist.gov/csrc/media/presentations/2025/stppa7-timelock-encryption/images-media/stppa7-timelock-encryption.pdf 30. Drand | Filecoin Docs, https://docs.filecoin.io/basics/the-blockchain/drand 31. Filecoin Features: Distributed Randomness & Leader Elections, https://filecoin.io/blog/posts/filecoin-features-distributed-randomness-leader-elections/ 32. Building Random, Fair, and Verifiable Games on Blockchain. Raffle smart contract designs on Sui Network - arXiv, https://arxiv.org/pdf/2310.12305 33. Verifiable Random Functions: Fair Play in a Decentralized World - Supra, https://supra.com/academy/verifiable-random-functions-fair-play-in-a-decentralized-world/ 34. Mempool Privacy via Batched Threshold Encryption: Attacks and Defenses - USENIX, https://www.usenix.org/system/files/usenixsecurity24-choudhuri.pdf