KausaLayer Resolver Network

A Trustless Resolution Layer for Prediction Markets on Solana

Fasqua

kausalayer.com

May 2026

Abstract

Prediction markets aggregate dispersed information into probabilistic forecasts about real-world events. Their utility, however, is bounded by the integrity of the resolution mechanism that maps event outcomes to on-chain payouts. Existing resolution systems rely either on centralized authorities or on optimistic oracles that depend on human voting and dispute periods, introducing latency, manipulation surface, and opacity into the settlement process. We present the KausaLayer Resolver Network (KRN), a resolution infrastructure for prediction markets on Solana that replaces human voting with cryptographically verifiable computation. KRN combines three primitives: (i) zero-knowledge transport-layer security (zkTLS) proofs that attest to the authenticity of data fetched from public web sources without trusting any intermediary, (ii) on-chain verification of these proofs through Groth16 pairing operations executed inside Solana's runtime, and (iii) zero-knowledge ownership proofs that allow winning bettors to claim payouts to fresh addresses without revealing their original betting identity. Outcomes are determined by deterministic execution of structured rules within a zero-knowledge virtual machine, eliminating the need for natural-language interpretation. We describe the system's architecture, the cryptographic constructions that underpin its security, and the threat model under which it operates. We argue that resolution by deterministic rule execution over cryptographically authenticated data offers a foundation for prediction markets that is faster, more transparent, and more resistant to manipulation than systems based on optimistic adjudication.

Keywords: prediction markets, zero-knowledge proofs, zkTLS, Solana, Groth16, oracles, trustless resolution

1. Introduction

A prediction market is a venue in which participants trade contracts whose payouts are contingent on the outcome of a future event. The market price of a contract is widely interpreted as a probabilistic forecast of the underlying event. The literature on information aggregation in markets, beginning with the seminal work of Hayek and extended by Wolfers and Zitzewitz, establishes that prediction markets can produce forecasts that are competitive with or superior to those of expert panels and statistical

Archived copy · cryp2nova.com

models. Their accuracy, however, is conditional on a resolution process that faithfully translates real-world outcomes into contractual payouts.

Resolution is the operation by which a prediction market terminates: an event occurs, an authoritative outcome is determined, and the on-chain state transitions such that holders of winning positions are entitled to claim. The integrity of this operation is not auxiliary to the market; it is constitutive of it. A market whose resolution is slow, contestable, or manipulable cannot reliably aggregate information, because rational participants will discount the expected payout by the perceived probability of resolution failure. The market price ceases to be a clean estimator of the event probability and becomes contaminated by estimates of resolver behavior.

Existing decentralized prediction markets resolve through one of two mechanisms. The first is a centralized authority, typically the platform operator, who declares outcomes by fiat. This approach is fast and cheap but reintroduces the trust assumption that decentralized systems are designed to eliminate. The second is an optimistic oracle, of which the Universal Market Access protocol (UMA) is the canonical instance. Under this scheme, a proposer asserts an outcome along with a bond; if no party disputes the assertion within a fixed window, the outcome is accepted; if a dispute is raised, a vote among token holders determines the final outcome. The optimistic oracle approach is more decentralized than fiat resolution but introduces three structural weaknesses: it requires a dispute window during which the market cannot settle, it depends on the honesty of a majority of voters, and it remains vulnerable to vote-buying and governance attacks against the resolving token.

This paper presents an alternative. The KausaLayer Resolver Network (KRN) is a resolution layer that replaces voting with verifiable computation. Outcomes are determined by executing structured rules over data that has been cryptographically authenticated through zkTLS proofs. The proofs are verified on-chain by the Solana runtime through alt_bn128 pairing precompiles. Bettor privacy is preserved through commitment-based betting and Groth16 ownership proofs. No human vote is required, no dispute window is necessary, and the trust assumption is reduced to the soundness of the underlying cryptographic primitives and the availability of the data sources named in the market specification.

The remainder of this paper is organized as follows. Section 2 surveys related work in oracle design and trustless data attestation. Section 3 describes the threat model and design objectives. Section 4 presents the system architecture. Section 5 details the cryptographic constructions. Section 6 specifies the resolution protocol. Section 7 addresses bettor privacy. Section 8 analyzes security. Section 9 examines limitations and discusses extensions left to future work. Section 10 concludes.

2. Related Work

2.1 Oracle Networks

The problem of bringing off-chain data on-chain has motivated a substantial literature and a number of production systems. Chainlink operates a federated network of independent node operators who report

Archived copy · cryp2nova.com

data from external sources, with consensus achieved through median or weighted aggregation. Pyth Network specializes in low-latency price data, sourced directly from a permissioned set of institutional publishers. These systems trade trustlessness for latency: they provide near-real-time data but require trust in the honesty of their reporters, with security guarantees grounded in economic stake and reputational incentives rather than cryptographic verification.

2.2 Optimistic Oracles

Optimistic oracle protocols, exemplified by UMA, accept any proposed outcome by default but allow disputes during a fixed window. Disputes are resolved by token-holder votes. This design assumes a rational majority of voters and a sufficient distribution of voting power; both assumptions have been challenged in practice. In March 2025, a coordinated voting attack on UMA's Data Verification Mechanism resulted in the incorrect resolution of a market with substantial value at stake, demonstrating that the security properties of optimistic systems are bounded by the political economy of their voting tokens.

2.3 zkTLS and Verifiable Web Data

The challenge of proving that data was obtained from a specific web endpoint, without trusting any party that observed the connection, has been addressed by zero-knowledge transport-layer security (zkTLS) protocols. TLSNotary and DECO were early academic constructions; more recent implementations include Reclaim Protocol, which generates proofs through an attestor network, and zero-knowledge virtual machine (zkVM) approaches such as those employed by Succinct's SP1, which replay TLS sessions inside a zero-knowledge proof system. The latter approach is of particular interest because it removes the need for a trusted attestor network: the entirety of the TLS handshake, certificate validation, and response decryption is performed inside the zkVM, producing a succinct proof that the named server returned the named data.

2.4 Zero-Knowledge Proofs on Solana

Solana provides native runtime support for the alt_bn128 pairing curve, exposing precompiles for elliptic curve addition, scalar multiplication, and pairing checks. This enables on-chain verification of Groth16 proofs at compute costs that are practical for routine use. Several Solana-native libraries, including groth16-solana, expose this functionality to programs written in Anchor or Pinocchio, allowing application-layer protocols to verify zero-knowledge proofs without trust in any external verifier.

3. Threat Model and Design Objectives

3.1 Adversarial Capabilities

We consider adversaries who may control any subset of the following parties.

Archived copy · cryp2nova.com

The market creator. A party who initializes a market and specifies its resolution rules. The market creator may be adversarial in its choice of rules, but cannot retroactively alter rules once the market is opened.

Bettors. Any party may participate in the market by placing commitments. An adversarial bettor may attempt to claim winnings to which they are not entitled, double-claim, or correlate their claims with their original betting addresses.

Proof submitters. Any party may submit a zkTLS proof to resolve a market. Submitters may be adversarial; they may attempt to submit invalid proofs, proofs for incorrect markets, or proofs that misreport the outcome.

Network observers. Any party may observe on-chain state. Observers may attempt to deanonymize bettors by correlating commitment, claim, and recipient addresses.

We do not consider adversaries who break the underlying cryptographic primitives: the discrete logarithm problem on the alt_bn128 curve, the collision resistance of Poseidon and SHA-256, or the soundness of the Groth16 proving system. We assume the certificate authorities trusted by the named data sources are themselves honest, in line with the standard web TLS trust model.

3.2 Design Objectives

KRN is designed to satisfy the following objectives.

Trustlessness. No human or institutional intermediary should be required to determine the outcome of a market. Outcomes should follow deterministically from the structured rules specified at market creation and the data returned by the named sources at resolution time.

Permissionless participation. Any party should be able to create markets, place bets, submit proofs, and claim winnings, subject only to the cryptographic and economic constraints of the protocol.

Verifiability. Every step of the resolution process should produce on-chain artifacts that can be independently verified by any observer. There should be no opaque off-chain computation whose result is trusted on faith.

Bettor privacy. A bettor's claim of winnings should not be linkable on-chain to their original betting address.

Resilience to source failure. A market should be resolvable even if individual data sources are temporarily unavailable, provided a quorum of named sources remains accessible.

4. System Architecture

KRN is composed of an on-chain Solana program written using the Anchor framework, an off-chain proof generation pipeline based on Succinct's SP1 zkVM, and a circuit for ownership proofs

Archived copy · cryp2nova.com

implemented in Circom and verified through Groth16 on the alt_bn128 curve.

4.1 On-Chain Component

The on-chain program manages the lifecycle of each market through a state machine consisting of five states: Open, Closed, Resolving, Resolved, and Failed. State transitions are effected by instructions that the program exposes to clients. The program maintains, for each market, an account containing the market identifier, the source configurations against which proofs will be validated, the resolution threshold and comparison operator, and a Poseidon Merkle tree of bet commitments.

Bet commitments are hashes of a tuple consisting of the market identifier, the chosen outcome, the bet amount, a secret nonce, and the bettor's public key. The Poseidon Merkle tree is updated incrementally as each bet is placed, and its root constitutes a public commitment to the set of bets accepted by the market. This root is later used as a public input to the ownership proof, allowing bettors to prove membership in the commitment set without revealing the specific commitment they hold.

4.2 Off-Chain Proof Generation

When a market reaches its resolution time, any party may generate a zkTLS proof attesting to the outcome. The proof generation process operates as follows. The prover establishes a TLS session with the named data source, recording the full handshake transcript including server certificates, ciphertext, and the prover's own session keys. This transcript is supplied as input to a guest program executing within the SP1 zkVM. Inside the zkVM, the TLS session is replayed: the certificate chain is validated against the WebPKI root of trust, the session keys are derived from the recorded handshake, and the response ciphertext is decrypted. The plaintext response is then parsed to extract the relevant numeric value, and the value is compared against the threshold specified in the market configuration to determine the outcome. The guest program commits, as public output, the outcome bit, the threshold, the extracted value, and the server name, along with the original zkTLS output for backward compatibility.

The SP1 zkVM produces a succinct Groth16 proof that the guest program executed correctly with the committed outputs. This proof is then submitted on-chain.

4.3 On-Chain Verification

The on-chain program verifies the submitted proof through a sequence of checks. First, the proof's verification key hash is checked against the expected hash for the KRN guest program, ensuring that the proof was generated by the correct circuit. Second, the Groth16 proof is verified through the alt_bn128 pairing precompiles. Third, the structured public values are parsed to extract the outcome, threshold, and server name. Fourth, the threshold and comparison operator are checked against the values stored in the market account, and the server name is hashed and checked against the source configuration. If all checks pass, the proof is recorded as a valid submission for the corresponding source index, and the market's source count is incremented.

Archived copy · cryp2nova.com

4.4 Resolution and Claim

Once the market has accumulated proofs from a configured minimum number of sources, any party may invoke the aggregate resolution instruction. This instruction tallies the outcomes reported across all submitted proofs and, if a strict majority is observed, finalizes the market with that outcome. The market state transitions to Resolved.

A winning bettor may then claim their payout by submitting a Groth16 ownership proof. The proof attests that the bettor knows a commitment in the on-chain Merkle tree whose recorded outcome matches the resolved outcome, without revealing which commitment. The proof additionally asserts the correctness of a nullifier, which is recorded on-chain to prevent double-claims, and is bound to a recipient address that need not have any prior relationship to the bettor's betting address. Upon successful verification, the program transfers the proportional payout from the market pool to the recipient.

5. Cryptographic Constructions

5.1 zkTLS Proof System

KRN's zkTLS proof system is built on Succinct's SP1, a zkVM that compiles RISC-V programs into arithmetic circuits and produces succinct Groth16 proofs of their execution. The advantages of this approach over earlier zkTLS schemes are twofold. First, the entire TLS verification logic, including certificate chain validation, can be expressed in standard Rust and compiled to the zkVM target without manual circuit construction. Second, no trusted attestor network is required; the proof is self-contained and can be verified by any party who possesses the verification key.

The proof's verification key is committed to in the on-chain program as a constant. This commitment binds the on-chain protocol to a specific guest program: a proof generated by a different program, even with identical claimed outputs, will not verify. Updates to the guest program therefore require a corresponding program upgrade, and constitute a governance event subject to the program's upgrade authority controls.

5.2 Poseidon Merkle Tree

The on-chain commitment tree is constructed using the Poseidon hash function over the BN254 scalar field. Poseidon was chosen because it is the de facto standard for hashing in arithmetic circuits, and because Solana's runtime exposes a native sol_poseidon syscall that performs the hash without the stack overhead that pure Rust implementations of Poseidon incur on the BPF target. The tree has a fixed depth of ten, supporting up to 1024 commitments per market.

The tree is incrementally maintained: when a new commitment is inserted at index i, the program walks from leaf to root, hashing each level's stored value with either the new value or a precomputed zero hash, depending on the parity of i at that level. Only the path from leaf to root is touched, requiring O(log n) hashes per insertion. The resulting root is stored in the market account and used as a public

Archived copy · cryp2nova.com

input to subsequent ownership proofs.

5.3 Ownership Circuit

The ownership circuit, implemented in Circom 2 and proven through Groth16 on the alt_bn128 curve, takes the following public inputs: the market identifier (reduced modulo the BN254 scalar prime), the resolved outcome, a nullifier, the commitment Merkle root, and the bet amount. Its private inputs are the secret nonce, the original bettor's public key represented as a field element, the Merkle siblings along the path from the leaf to the root, and the direction bits indicating left-or-right at each level.

The circuit performs three classes of checks. First, it reconstructs the commitment by hashing the market identifier, outcome, amount, secret nonce, and original public key under Poseidon. Second, it verifies that this commitment is included in the Merkle tree whose root was provided as a public input, by walking the path from leaf to root and asserting equality with the public root. The direction bits at each level are constrained to be boolean to prevent malleability attacks. Third, it computes the nullifier as the Poseidon hash of the market identifier, secret nonce, and original public key, and asserts equality with the public nullifier. The outcome is also constrained to be boolean.

The circuit has a fixed structure that is independent of the bettor's identity, and the resulting proof reveals nothing about the bettor's commitment beyond what is implied by the public inputs.

5.4 Field Element Reconciliation

A subtlety arises from the difference between the on-chain market identifier, which is a 32-byte SHA-256 hash, and the circuit's representation of the same identifier as an element of the BN254 scalar field. The scalar field is approximately 2 to the 254 in size, while the SHA-256 output ranges over 2 to the 256, so a market identifier may exceed the field modulus. The Circom compiler reduces all inputs modulo the field prime automatically, but the on-chain validation logic must also reduce the stored market identifier before comparing it to the proof's public input. KRN's verifier performs this reduction explicitly, using a constant-time subtraction loop bounded by the ratio of 2 to the 256 to the field modulus.

6. Resolution Protocol

The resolution protocol proceeds in four phases.

Initialization. A market creator invokes the init_market instruction, supplying the market identifier, source configurations, threshold, comparison operator, close timestamp, and resolution deadline. The program creates the corresponding market account and initializes its state to Open.

Betting. While the market is Open, bettors invoke place_bet, supplying their commitment hash, chosen side, and bet amount. The program transfers the lamports to the market's pool, records the commitment in the bettor's commitment account, and inserts the commitment into the Merkle tree.

CRYP2NOVA

Archived copy · cryp2nova.com

Closing. After the close timestamp, any party may invoke close_market to transition the market to Closed. From this point, no further bets are accepted.

Proof Submission and Aggregation. Submitters may invoke submit_proof for any source index in the market's configuration. Each successful submission increments the source count and, on the first submission, transitions the market to Resolving. When the source count reaches the configured minimum, any party may invoke aggregate_resolution to tally the outcomes. If a strict majority of proofs report the same outcome, the market transitions to Resolved with that outcome.

Claim. Holders of winning commitments invoke claim_winning, supplying an ownership proof and a recipient address. The program verifies the proof, records the nullifier to prevent replay, computes the proportional payout, and transfers it to the recipient.

If the resolution deadline passes without a sufficient number of proofs, the market transitions to Failed, and bettors may invoke refund_market to recover their stakes.

7. Bettor Privacy

KRN's privacy model decouples a bettor's on-chain identity at the time of betting from their identity at the time of claiming. The mechanism is as follows.

When placing a bet, a bettor generates a secret nonce uniformly at random. Their commitment is the Poseidon hash of the market identifier, the chosen outcome, the bet amount, the nonce, and a public key of their choice. This commitment is recorded on-chain along with the side and amount; the side and amount are visible because they are required for pool accounting, but the commitment hash conceals the nonce and the public key under the cryptographic security of Poseidon.

When the market resolves and the bettor wishes to claim, they generate an ownership proof using the nonce and public key as private inputs, and a recipient address of their choice as the destination for the payout. The recipient address need not have any prior on-chain history with the bettor; it can be a fresh keypair generated for the purpose of receiving the payout. The nullifier, computed inside the circuit, is bound to the original public key and the secret nonce, ensuring that each commitment can be claimed at most once. From the perspective of an on-chain observer, the bet and the claim share only the market identifier and the amount; there is no direct cryptographic link between the bettor's pre-bet activity and their post-claim recipient.

The privacy guarantees are bounded by the practical anonymity set of bettors in the same market with the same amount and outcome. A market with few participants will offer a smaller anonymity set than a market with many. Bettors who require strong privacy should select markets with substantial participation.

8. Security Analysis

8.1 Soundness of Resolution

Archived copy · cryp2nova.com

The outcome of a resolved market is the value asserted by a strict majority of submitted proofs. For an adversary to force an incorrect resolution, the adversary must convince the program to accept a sufficient number of proofs reporting the false outcome. Each proof's public values are bound to the verification key of the KRN guest program; an adversary cannot simply construct arbitrary public values. To produce a proof with a false outcome, the adversary must either break the soundness of the SP1 proof system or compromise one of the named data sources at the time of resolution. The former is excluded by our cryptographic assumptions. The latter is mitigated by the multi-source requirement: a market with k named sources requires the adversary to compromise at least the ceiling of (k+1)/2 of them simultaneously to influence the outcome. Increasing the number of named sources increases the adversary's cost linearly.

8.2 Soundness of Claims

The ownership circuit's soundness ensures that no party can produce a valid proof unless they know the secret nonce of a commitment in the Merkle tree whose recorded outcome matches the resolved outcome. The nullifier construction prevents the same commitment from being claimed twice: the program records nullifiers in account storage and rejects proofs whose nullifier is already present. The binding between the nullifier and the original public key, which is itself bound by the commitment hash, prevents an adversary from claiming on behalf of a commitment they did not place.

8.3 Protocol-Level Attacks

Several protocol-level concerns deserve treatment.

Front-running of proofs. Because proof submission is permissionless, an adversary may attempt to submit a proof reporting an incorrect outcome before honest provers can submit correct proofs. This attack is defeated by the multi-source requirement and the majority aggregation rule: an isolated incorrect proof cannot resolve the market against a quorum of correct proofs.

Source manipulation. A market creator who specifies an attacker-controlled source as one of the required sources may, in concert with the source operator, influence the resolution. This is a market-creation-time concern, not a protocol-level vulnerability; bettors must inspect the named sources before participating in a market and decline to participate in markets that name untrustworthy sources.

Proof generation cost. An adversary may attempt to delay resolution by withholding proofs. Because proof generation is permissionless, this attack requires the adversary to suppress all honest provers, which is increasingly costly as the prover community grows. The fallback path, in which a market is refunded if the resolution deadline passes without sufficient proofs, ensures that bettors cannot have their stakes locked indefinitely.

8.4 Privacy Adversaries

A network observer may attempt to correlate a claim with a bet through statistical inference: amount, timing, gas usage, and other side channels. The privacy guarantees of KRN are with respect to the

Archived copy · cryp2nova.com

on-chain cryptographic state, not with respect to all possible side-channel observations. Bettors who require privacy against well-resourced adversaries should adopt operational practices to mitigate side channels, including using fresh recipient addresses, varying claim timing, and routing transactions through privacy-preserving relays.

9. Limitations and Future Work

The current design has several limitations that motivate continued research and development.

Subjective outcomes. KRN's resolution model is grounded in the deterministic execution of structured rules over web-accessible data. Markets whose outcomes depend on subjective judgment, on data that is not available through any web endpoint, or on events for which no authoritative source exists are outside the scope of the present design. Such markets would require a hybrid mechanism that combines structured rules where applicable with a fallback to human adjudication where necessary.

Source availability. The protocol assumes that named data sources will be reachable at the time of resolution. Sources may be permanently removed, may change their response format, or may impose access restrictions that prevent zkTLS proof generation. The refund mechanism handles total source failure but does not address gradual format drift or partial outages. A more sophisticated source management layer, including reputation tracking and dynamic source selection, is left to future work.

Proof generation latency. The zkTLS proof generation process is computationally intensive and requires specialized hardware to achieve acceptable latency. The current design assumes a permissionless community of provers will emerge; an explicit incentive layer to reward timely proof generation, and a network of decentralized provers, would strengthen the protocol's liveness guarantees.

Circuit upgrades. The on-chain verification key is hardcoded into the program. Upgrading the guest program requires upgrading the on-chain program, an event that is subject to the program's upgrade authority. A transition to immutable program code, or to a verification key registry that supports controlled rotation, would reduce the trust placed in the upgrade authority over the long term.

Field element representation. The reconciliation of SHA-256 market identifiers with BN254 field elements requires a modular reduction at verification time. While this reduction is constant-time and bounded, an alternative design that uses a hash function whose output is already a field element would simplify the verification logic and reduce the attack surface.

Anonymity set. The privacy guarantees of the ownership circuit are bounded by the size of the anonymity set in each market. Cross-market anonymity, in which bettors across multiple markets are indistinguishable, is not a property of the present design and would require additional cryptographic machinery, likely involving privacy pools or shared commitment trees.

Generalized rule expressiveness. The current guest program supports threshold comparison over numeric values extracted from JSON responses. A more general rule language, supporting set membership, range queries, multi-source aggregation within a single proof, and HTML parsing, would

Archived copy · cryp2nova.com

expand the class of markets that can be resolved by structured execution.

10. Conclusion

The integrity of a prediction market is determined by the integrity of its resolution mechanism. Existing approaches to resolution either reintroduce trust assumptions that decentralization is meant to eliminate, or rely on optimistic adjudication that exposes the market to dispute latency and governance attacks. We have presented the KausaLayer Resolver Network, a resolution layer that replaces voting with verifiable computation. By combining zero-knowledge transport-layer security proofs, on-chain verification through pairing-based cryptography, and zero-knowledge ownership proofs that preserve bettor privacy, KRN reduces the trust assumptions of prediction market resolution to the soundness of well-studied cryptographic primitives and the availability of web data sources named at market creation. Outcomes follow deterministically from rules and data; no human vote intervenes; no dispute window is required. We believe this approach offers a path toward prediction markets that are not only faster and more transparent than the current generation of decentralized markets, but also more resistant to the political-economic attacks that have repeatedly compromised optimistic oracles.

Future work will extend KRN to richer rule languages, decentralized prover networks, and stronger anonymity sets. The core insight, however, is that the resolution of an objective market need not depend on human judgment: the rules of resolution can be structured, the data of resolution can be cryptographically authenticated, and the verification of resolution can be performed by mathematics alone.

References

[1] F. A. Hayek. "The Use of Knowledge in Society." *American Economic Review*, 35(4):519–530, 1945.

[2] J. Wolfers and E. Zitzewitz. "Prediction Markets." *Journal of Economic Perspectives*, 18(2):107–126, 2004.

[3] J. Groth. "On the Size of Pairing-Based Non-Interactive Arguments." In *Advances in Cryptology — EUROCRYPT 2016*, LNCS 9666, pp. 305–326. Springer, 2016. DOI: 10.1007/978-3-662-49896-5_11.

[4] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger. "Poseidon: A New Hash Function for Zero-Knowledge Proof Systems." In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 519–535, 2021. ePrint: eprint.iacr.org/2019/458.

[5] F. Zhang, S. K. D. Maram, H. Malvai, S. Goldfeder, and A. Juels. "DECO: Liberating Web Data Using Decentralized Oracles for TLS." In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20)*, pp. 1919–1938. DOI: 10.1145/3372297.3417239.

[6] H. Ritzdorf, K. Wuest, A. Gervais, G. Felley, and S. Capkun. "TLS-N: Non-repudiation over TLS Enabling Ubiquitous Content Signing for Disintermediation." *NDSS Symposium*, 2018.

[7] Succinct Labs. "SP1: A Performant, 100% Open-Source, Contributor-Friendly zkVM." Available at github.com/succinctlabs/sp1, 2024.

[8] H. Lambur and A. Lu. "UMA: A Decentralized Financial Contract Platform." Risk Labs Foundation whitepaper, December 2018. uma.xyz/UMA-whitepaper.pdf.

Archived copy · cryp2nova.com

[9] The Block. "Polymarket says governance attack by UMA whale to hijack a bet's resolution is ‘unprecedented’." March 27, 2025. theblock.co/post/348171.

[10] Solana Foundation. "Native Programs: alt_bn128 Compression and Pairing." *Solana Documentation*. solana.com/docs/core/programs.

[11] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. "Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture." In *23rd USENIX Security Symposium (USENIX Security 14)*, pp. 781–796, 2014.

This document describes the design of the KausaLayer Resolver Network. The design is subject to refinement as the system is further developed and as feedback from the cryptographic and prediction market communities is incorporated.

KausaLayer

Privacy infrastructure on Solana

kausalayer.com

CRYP2NOVA

Archived copy · cryp2nova.com