© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716368

VeriTrust: An AI-Powered Decentralized Reputation System for the Gig Economy

MUIZ ZATAM[1], ANGAD MUTHYALA[2], SARVESH VARVATKAR[3], JASPREET KAUR[4]
[1,2,3,4] *Dept. of Computer Engineering, Smt. Indira Gandhi College of Engineering, Navi Mumbai, India*

Abstract- ***Trust in the gig economy lives and dies by reputation. Platforms like Upwork and Fiverr use review scores as stand-ins for credibility, yet the systems underlying those scores are opaque, siloed, and surprisingly easy to manipulate. This paper presents VeriTrust, a fully working AI-powered reputation system that pairs blockchain immutability with natural language processing to give freelancers a tamper-proof, portable reputation they genuinely own. At its core, VeriTrust runs an application-level Proof-of-Work blockchain with SHA-256 mining and RSA-2048 cryptographic signing so that every review is verifiably authentic. Two Solidity smart contracts are deployed on Polygon Amoy—one storing full data for maximum auditability, and a leaner VeriTrustLite variant that trims gas costs by roughly 80% through struct packing, custom errors, and hash-only storage. A three-step prepare/confirm write protocol ties each on-chain anchor to the matching PostgreSQL record, and reputation is keyed on RSA public keys rather than wallet addresses, so users can carry their history across chains and platforms without asking anyone's permission. A React.js DApp with MetaMask integration, a live blockchain explorer, and a chain indexer round out the implementation. Smart contract test coverage is 100% across all statement branches.***

Index Terms— Blockchain, Decentralized Application, Gig Economy, Natural Language Processing, Proof-Of-Work, Reputation Systems, RSA Cryptography, Sentiment Analysis, Smart Contracts.

I. INTRODUCTION

The gig economy has grown remarkably fast over the past decade. Platforms such as Upwork, Fiverr, and Freelancer now connect tens of millions of freelancers with clients worldwide, spanning a labor market worth hundreds of billions of dollars annually. Within these ecosystems, reputation scores and client reviews serve as the primary proxies for trust—shaping hiring decisions, justifying price premiums, and ultimately determining how much a freelancer can earn [8].

The problem is that existing reputation mechanisms are centralized in every important sense. Platforms act as sole custodians of reputation data: they own it, they can delete or suppress reviews quietly, and they keep it locked inside proprietary silos. A freelancer who spends years building credibility on Fiverr cannot carry that capital over to a competing platform [1]. Meanwhile, the absence of intelligent validation means fake reviews, coordinated Sybil attacks, and AI-generated spam remain persistent threats that current moderation pipelines fail to neutralize reliably [6].

This paper introduces VeriTrust, an end-to-end implemented system that tackles these limitations through four integrated components: (1) an application-level PoW blockchain with SHA-256 mining, RSA-2048 per-transaction signing, and JSON file persistence; (2) two Solidity smart contracts on Polygon Amoy; (3) a hybrid storage architecture using a three-step prepare/confirm protocol; and (4) a React/Node.js DApp with MetaMask integration, live blockchain explorer, chain indexer, and NLP-based review validation. The system achieves 100% smart contract branch coverage across 18 test cases.

II. RELATED WORK

A. Reputation as a Service

Hillebrand and Coetzee [1] introduced the Reputation-as-a-Service (RaaS) model, making a compelling case for API-first architectures that decouple reputation scores from individual platforms. Their work laid the conceptual groundwork for reputation portability but did not address tamper-proofing, cryptographic signing, or decentralized identity—gaps that VeriTrust directly fills.

B. Blockchain-Based Trust

Nguyen et al. [2] proposed a hybrid on-chain/off-chain trust framework for DApps, showing that anchoring critical proofs on-chain while processing data off-chain strikes a reasonable balance between cost, privacy, and security. That hybrid intuition directly motivates VeriTrust's dual-storage design. Zheng et al. [3] surveyed consensus mechanisms and layer-2 scaling, providing architectural patterns that shaped our contract design choices.

C. AI-Enhanced Review Validation

Hassan and Islam [4] showed that NLP-based sentiment features can detect fake reviews with high precision, establishing the methodologies VeriTrust's validation layer builds on. Kotha [5] extended this line of work to distributed real-time settings. Yu et al. [6] applied graph neural networks to detect coordinated review campaigns by modeling user-review-item interaction graphs.

D. Blockchain Review Storage and Gap Analysis

Martens and Maalej [7] demonstrated with ReviewChain that immutable review anchoring via smart contracts is practically feasible. Bhatia et al. [8] proposed WorkerRep for crowdsourcing labor, giving workers ownership of immutable reputation records. Toxtli et al. [9] investigated fairness-aware review scoring. No prior work combines immutability, AI validation, portability, cryptographic signing, and a full implementation in a single runnable system—a gap VeriTrust addresses.

III. PROBLEM STATEMENT

Let F = {f1,...,fn} be the set of freelancers and C = {c1,...,cm} the set of clients on a gig platform P. A reputation score ρ(fi, P) is computed by P from a review multiset R(fi, P) where each review rj = (ratingj, commentj, reviewerj). Centralized platforms suffer from four compounding failure modes: (1) Opacity—the scoring function is undisclosed and unauditable; (2) Mutability—the platform can delete or modify reviews without notification; (3) Isolation—reputation built on one platform cannot be transferred to another; and (4) Fraud vulnerability—adversaries can inject fake reviews indistinguishable without NLP-level analysis. VeriTrust addresses all four: on-chain storage replaces the opaque datastore; RSA signatures replace platform-controlled authentication; a public API enables portability; and NLP validation gates every write.

IV. SYSTEM ARCHITECTURE

E. Five-Layer Overview

VeriTrust is organized into five discrete layers. L1 (UI) is the React.js DApp—all user interactions originate here, with MetaMask mediating wallet authentication and on-chain transaction signing. L2 (API) is the Node.js/Express application server hosting 15 REST endpoints, validating JWT tokens, performing RSA signature verification on all write operations, and running the Chain Indexer service. L3 (AI) is the NLP validation pipeline that preprocesses review text, analyzes sentiment polarity, and scores authenticity; only reviews passing the threshold are allowed through. L4 (Storage) is the dual-store: PostgreSQL for human-readable data, ChromaDB for embedding-based similarity queries, and the PoW Engine for block mining. L5 (Blockchain) is the immutable anchor layer comprising the application-level PoW chain and the Polygon Amoy smart contract.

F. Three-Step Write Protocol

All mutation operations follow a strict prepare/onchain/confirm protocol ensuring the on-chain anchor is always written before the off-chain record. Step 1 (Prepare): the frontend calls POST /api/:resource/prepare, which computes the keccak256 hash and returns it without writing anything. Step 2 (On-chain): the frontend sends only the hash to the Polygon smart contract via MetaMask; the transaction hash and block number come back. Step 3 (Confirm): the frontend calls POST /api/:resource/confirm with the full data plus the transaction receipt; the backend writes the complete record to PostgreSQL and sets verification_status = verified. This design cryptographically binds every PostgreSQL record to its on-chain anchor, making silent tampering immediately detectable.

V. AUTHENTICATION AND IDENTITY MODEL

© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716368

VeriTrust uses RSA-2048 PEM public keys as the primary identity anchor rather than Ethereum wallet addresses. On registration, the browser generates a key pair via Node.js crypto.generateKeyPairSync; the private key is stored locally and never transmitted. Because reputation records are keyed on the RSA public key, a user who moves to a different Ethereum wallet, chain, or platform can prove ownership of their reputation by presenting the same key pair. For session authentication, users submit their RSA private key to POST /login; the backend validates it by a test sign-and-verify round trip, derives the public key, and issues a 24-hour JWT. All write operations additionally require per-transaction RSA signatures on a deterministically serialized canonical payload with a timestamp anti-replay field, ensuring signature mismatches yield an immediate 401 rejection.

VI. APPLICATION-LEVEL BLOCKCHAIN

Each call to addTransaction(payload) mines a new block containing exactly one typed transaction payload. The mining loop increments a nonce until the SHA-256 hash of the block header begins with d=2 leading zero hex digits, yielding a median mining time under 3 ms and a 95th-percentile below 10 ms on commodity hardware. The block is appended to db/chain.json via an atomic write. On startup, the entire chain is re-validated: each block's hash is recomputed and previousHash linkage is checked; any discrepancy prevents the server from starting. The chain recognizes five transaction types: REGISTER_PROFILE, UPDATE_PROFILE, ADD_SERVICE, POST_REVIEW, and REVIEW_RESPONSE. Query functions always return the latest state by walking forward through the chain, pre-computing per-freelancer average ratings for the marketplace browse endpoint.

VII. SMART CONTRACT DESIGN

G. VeriTrust.sol: Full On-Chain Storage

The first contract (Solidity 0.8.24) stores complete profile and review data on-chain for maximum auditability. Identity is keyed on RSA-2048 PEM public key strings rather than msg.sender. The Review struct stores: subject key, reviewer key, reviewer name, rating (uint8, 1–5), comment (max 1000 chars), RSA signature (base64), verified boolean, timestamp, and response fields. Five events (ProfileRegistered, ProfileUpdated, ServiceAdded, ReviewPosted, ReviewResponsePosted) feed the chain indexer. All state-modifying functions include defensive require guards with descriptive revert strings.

H. VeriTrustLite.sol: Gas-Optimized Variant

VeriTrustLite.sol replaces string storage with bytes32 keccak256 hashes and applies four optimization techniques: (1) struct packing—co-locating boolean and small-integer fields reduces the Review struct from 30+ storage slots to 4; (2) custom errors replace require strings, saving ~200 gas per revert; (3) address-based auth uses msg.sender as the identity key; and (4) calldata parameters replace memory on all external functions. This contract is deployed on Polygon Amoy as the production variant, achieving approximately 80% gas savings across all operations (register ~65k vs ~320k gas units, post review ~72k vs ~380k).

VIII. HYBRID STORAGE AND FRONTEND

I. PostgreSQL Schema and Chain Indexer

The off-chain schema defines four primary tables (wallets, profiles, services, reviews), each carrying an integrity bridge: content_hash (keccak256 of the record's data fields), tx_hash (Polygon transaction hash), block_number, and verification_status (pending / verified / tampered). ChainIndexer.js instantiates an ethers.js JsonRpcProvider pointed at Polygon Amoy and subscribes to all five contract events. For each ReviewPosted event, it performs an off-chain RSA re-verification pass. The indexer acts as an independent, continuously-running auditor that catches discrepancies between on-chain state and off-chain data without any centralized coordinator.

J. React Frontend

The React.js frontend (React 18, Vite bundler, GSAP for scroll animations) delivers a marketplace-grade experience across seven pages: Home, Freelancers directory, Gig Page, Write Review (with live NLP sentiment preview), Profile, Explorer (live blockchain explorer), and Dashboard (analytics over reputation trends). All blockchain interactions are centralized in blockchainService.js, which implements a dual-read strategy: it first queries contract events from the Polygon Amoy RPC and falls back to PostgreSQL

data if the RPC is unavailable, ensuring the explorer always renders useful information.

IX. REPUTATION SCORING MODEL

The reputation score for freelancer f is a confidence- and recency-weighted average over verified reviews. The per-review weight wr integrates NLP authenticity confidence conf(r) with temporal decay: wr = α · conf(r) + (1−α) · e^(−λ·Δt), where α=0.6 and λ=0.002 (half-life ~347 days). Anonymous reviews receive a fixed confidence penalty of conf(r)=0.4 regardless of NLP output. The live platform currently exposes the simpler arithmetic mean as averageRating in the profile API response; the full weighted model is applied in the reputation dashboard once NLP confidence scores are available from the validation layer.

X. EVALUATION

K. Smart Contract Test Coverage

The Hardhat test suite (VeriTrust.test.js) covers all contract functions across 18 test cases, including positive paths, boundary conditions, and revert scenarios. Tested invariants include: duplicate registration reverts; ratings outside 1–5 revert; reviews for nonexistent profiles revert; empty reviewer names default to "Anonymous"; RSA signatures are stored faithfully; duplicate seller responses revert; and profile count increments correctly. The suite achieves 100% coverage across statements, branches, functions, and lines for all four test groups: Profile Management, Service Management, Review Management, and View Functions.

L. Performance and Gas Analysis

PoW mining at d=2 over 500 blocks on an Intel Core i7 workstation yields a median mining time of ~3 ms and a 95th percentile below 10 ms, confirming that the application-level chain introduces no latency bottlenecks. Gas cost analysis using the Hardhat gas reporter shows VeriTrustLite.sol achieves approximately 80% savings compared to VeriTrust.sol: register profile (~65k vs ~320k), add service (~58k vs ~280k), post review (~72k vs ~380k), and post response (~38k vs ~140k).

M. Security Analysis

Sybil attack resistance is achieved by requiring RSA-2048 key ownership and a valid per-transaction signature for verified reviews, with anonymous reviews requiring PoW puzzle solving. Replay attacks are prevented by a timestamp field with a configurable acceptance window. Off-chain tampering detection is continuous: any PostgreSQL data modification produces a keccak256 hash mismatch against the on-chain anchor, triggering verification_status = tampered in the chain indexer. All require guards and custom errors in the contracts prevent duplicate profiles, invalid ratings, out-of-bounds indices, and duplicate responses.

XI. CONCLUSION AND FUTURE WORK

This paper presented VeriTrust, a fully implemented AI-powered decentralized reputation system combining RSA cryptographic signing, an application-level PoW blockchain, two Solidity smart contracts on Polygon Amoy, NLP-based review validation, and a PostgreSQL hybrid-storage architecture into a single runnable DApp. Key contributions include: RSA-keyed portable identity enabling cross-chain and cross-platform portability without trusted intermediaries; a dual blockchain strategy combining low-latency local operation with globally auditable anchoring; a three-step prepare/confirm protocol making tampering continuously detectable; two contract variants achieving ~80% gas reduction through struct packing, custom errors, and hash-only storage; and 100% smart contract coverage across 18 test cases. Future work includes federated NLP training, ZK-SNARK proofs for private review attestation, threshold cryptography for key recovery, layer-2 rollup deployment, a standardized cross-platform reputation API, and DAO-based governance of moderation policies.

ACKNOWLEDGMENT

The authors thank Prof. Jaspreet Kaur (Internal Guide) and Dr. K. T. Patil (Head of Department, Computer Engineering) at Smt. Indira Gandhi College of Engineering, Navi Mumbai, for their guidance throughout this project.

REFERENCES

[1] C. Hillebrand and A. P. Coetzee, “Towards Reputation-as-a-Service in the Cloud,” in Proc. Int. Conf. Cloud Computing and Services Science (CLOSER), 2013, pp. 371–380.

[2] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, “Blockchain-based Trust System for Decentralised Applications: A Survey,” IEEE Access, vol. 9, pp. 118270–118299, 2021.

[3] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, “Blockchain-Based Decentralized Application: A Survey,” IEEE Trans. Syst., Man, Cybern.: Syst., vol. 53, no. 6, pp. 3725–3743, 2023.

[4] M. M. Hassan and M. S. Islam, “Impact of Sentiment Analysis in Fake Online Review Detection,” in Proc. Int. Conf. ICT for Sustainable Development (ICT4SD), 2021, pp. 673–683.

[5] S. Kotha, “Distributed Fake Review Detection and Real-Time Anomaly Detection Using Machine Learning,” J. Computational Intelligence, vol. 12, no. 1, pp. 45–62, 2025.

[6] D. Yu, Y. Zhang, Z. Huang, W. Wang, and P. S. Yu, “Graph Learning for Fake Review Detection,” in Proc. ACM Web Conf. (WWW), 2022, pp. 2879–2888.

[7] D. Martens and W. Maalej, “ReviewChain: Untampered Product Reviews on the Blockchain,” in Proc. IEEE Int. Conf. Software Architecture Companion (ICSA-C), 2018, pp. 126–129.

[8] M. Bhatia, A. Kumar, and D. Sangwan, “WorkerRep: Immutable Reputation System For Crowdsourcing Platform Using Blockchain,” in Proc. Int. Conf. Communication Systems & Networks (COMSNETS), 2020, pp. 748–753.

[9] C. Toxtli, A. Suri, and S. Savage, “Reputation Agent: Prompting Fair Reviews in Gig Markets,” in Proc. The Web Conf. (WWW), 2020, pp. 1343–1349.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in Proc. NAACL-HLT, 2019.

[11] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[12] V. Buterin, “A Next-Generation Smart Contract and Decentralized Application Platform,” Ethereum Whitepaper, 2014. [Online]. Available: https://ethereum.org