https://doi.org/10.51885/3107-2755_JISC_2026_2_4
MRNTI 20.53.19; 20.53.21

АРХИТЕКТУРА SELF-HOSTED СИСТЕМЫ BLOCKCHAIN FORENSICS ДЛЯ SOLANA: ПИЛОТНАЯ ОЦЕНКА РЕАЛИЗАЦИИ НА RUST

SOLANA ҮШІН SELF-HOSTED BLOCKCHAIN FORENSICS ЖҮЙЕСІНІҢ АРХИТЕКТУРАСЫ: RUST НЕГІЗІНДЕГІ ІСКЕ АСЫРУДЫ ПИЛОТТЫҚ БАҒАЛАУ

ARCHITECTURE OF A SELF-HOSTED BLOCKCHAIN FORENSICS SYSTEM FOR SOLANA: A PILOT EVALUATION OF A RUST IMPLEMENTATION

I. Kotlyarova [1], **A. Popov** [1], **T. Chernykh** [1]*, **K. Kirichyok** [1], **I. Petrov** [1],
[1] D. Serikbayev East Kazakhstan Technical University, Ust-Kamenogorsk, Kazakhstan
* Author-correspondent: Chernykh Timur, e-mail: rewvtrew@gmail.com

Ключевые слова:
Solana; blockchain forensics; Rust; анализ графов; обнаружение мошенничества; SPL-токены; MEV; кибербезопасность.

АННОТАЦИЯ
Высокая пропускная способность и многоуровневая модель данных Solana затрудняют применение методов blockchain forensics, разработанных для Bitcoin и Ethereum. Цель исследования — разработать архитектуру self-hosted системы blockchain forensics, реализованной на Rust, и выполнить первичную экспериментальную оценку её пригодности для сбора, нормализации и анализа транзакций Solana. Исследование объединяет тематический обзор с анализом прототипа, использующего read-only RPC, owner resolution для кошельков и SPL token accounts, временной мультиграф и объяснимые правила выявления wash trading, dusting, sandwich/MEV и pump-and-dump. Практическая оценка выполнена на наборе из 10 000 транзакций mainnet-beta для слотов 250 000 000–250 001 000 на сервере AMD EPYC 7532 с 64 ГБ памяти. Агрегированное время парсинга составило 38 мс для Rust и 3 800 мс для Python, что соответствует расчётным средним 0,0038 и 0,38 мс на транзакцию и стократному различию в условиях опыта. Построение графа заняло 21 и 2 100 мс, а пиковая память — 42 и 480 МБ соответственно. Поскольку реализации различались также числом потоков, результат характеризует сравниваемые программные конфигурации в целом и требует независимого воспроизведения. Научный вклад имеет системно-интеграционный характер и состоит в формализации проверяемого конвейера blockchain forensics для Solana.

Түйінді сөздер:
Solana; блокчейн-криминалистика; Rust; графтық талдау; алаяқтықты анықтау; SPL-токендер; MEV; киберқауіпсіздік.

ТҮЙІНДЕМЕ
Solana желісінің жоғары өткізу қабілеті және көпдеңгейлі деректер моделі Bitcoin мен Ethereum үшін әзірленген blockchain forensics әдістерін тікелей қолдануды қиындатады. Зерттеудің мақсаты — Rust тілінде іске асырылған self-hosted blockchain forensics жүйесінің архитектурасын әзірлеу және оның Solana транзакцияларын жинау, қалыпқа келтіру және талдау үшін жарамдылығын бастапқы эксперименттік бағалау. Жұмыс тақырыптық шолуды read-only

RPC, әмияндар мен SPL token accounts үшін owner resolution, уақыттық мультиграф және wash trading, dusting, sandwich/MEV мен pump-and-dump үлгілерін анықтауға арналған түсіндірілетін ережелерді пайдаланатын прототипті талдаумен біріктіреді. Практикалық бағалау AMD EPYC 7532 процессоры және 64 ГБ жады бар серверде mainnet-beta желісінің 250 000 000–250 001 000 слоттарынан алынған 10 000 транзакцияға жүргізілді. Жинақталған парсинг уақыты Rust үшін 38 мс және Python үшін 3 800 мс болды; есептік орташа мәндер бір транзакцияға тиісінше 0,0038 және 0,38 мс болып, осы тәжірибе жағдайында жүз есе айырмашылық көрсетті. Граф құру уақыты 21 және 2 100 мс, ал жадтың ең жоғары тұтынуы 42 және 480 МБ болды. Іске асырулар ағындар саны бойынша да ерекшеленгендіктен, нәтиже бағдарламалық конфигурацияларды тұтастай сипаттайды және тәуелсіз қайта тексеруді қажет етеді. Ғылыми үлес жүйелік-интеграциялық сипатта және тексерілетін blockchain forensics конвейерін қалыптастырудан тұрады.

keywords:
Solana; blockchain forensics; Rust; graph analysis; fraud detection; SPL tokens; MEV; cybersecurity.

ABSTRACT

Solana's high throughput and layered data model complicate the direct transfer of blockchain-forensics methods developed for Bitcoin and Ethereum. This study aims to design a self-hosted blockchain-forensics system implemented in Rust and provide a preliminary experimental assessment of its suitability for collecting, normalizing, and analysing Solana transactions. The research combines a thematic literature review with an assessment of a prototype based on read-only RPC access, explicit wallet-to-token-account owner resolution, a temporal multigraph, and explainable rules for wash trading, dusting, sandwich/MEV, and pump-and-dump patterns. The practical evaluation uses 10,000 mainnet-beta transactions from slots 250,000,000–250,001,000 on an AMD EPYC 7532 server with 64 GB RAM. Aggregate parsing times were 38 ms for Rust and 3,800 ms for Python, corresponding to calculated averages of 0.0038 and 0.38 ms per transaction and a 100-fold difference under the reported test conditions. Graph-construction times were 21 and 2,100 ms, while peak memory values were 42 and 480 MB, respectively. Because the implementations also differed in thread count, the result characterizes the compared software configurations as a whole and requires independent replication. The contribution is system-integrative: a testable blockchain-forensics pipeline for Solana that separates observed transaction facts, heuristic inferences, and an auditable evidence bundle.

INTRODUCTION

A public blockchain preserves transaction events in a ledger that can be independently verified. However, transparency does not turn an address into an identified person and does not remove ambiguity from the economic meaning of an operation. Funds may pass through intermediary wallets, token accounts, and smart contracts, while coordinated behaviour can be disguised as ordinary market activity. Consequently, a digital investigation requires more than viewing an explorer: raw data must be transformed reproducibly into relationship graphs, temporal features, and testable hypotheses (Meiklejohn et al., 2016; Conti et al., 2018).

The transition from Bitcoin to smart-contract platforms expanded the observable feature space. Ethereum and DeFi studies analyse contract events, exchange routes, pre- and post-operation balances, liquidations, economic cycles, and call ordering. These observations support methods for detecting Ponzi schemes, volume manipulation, pump-and-dump behaviour, and blockchain-extractable value (Chen et al., 2018; Bartoletti et al., 2020; Daian et al., 2020; Werner et

al., 2022). Direct transfer of those methods to Solana nevertheless requires architectural adaptation: instructions reference elements of account_keys, a token account is not identical to its owner wallet, one transaction may contain several programs and inner instructions, and Sealevel parallel execution changes the interpretation of event ordering (Solana Foundation, 2019; Alizadeh & Khabbazian, 2025).

The practical problem is intensified by throughput. A blockchain-forensics system must collect confirmed transactions without private keys, resolve the owners of SPL token accounts, retain failed operations, preserve amounts in atomic units, and maintain causal and temporal relationships between slots. Analysts must also understand why a rule fired: an opaque risk score without supporting facts limits both scientific verification and evidentiary value. A systematic review of blockchain forensics identifies fragmented data, the absence of common benchmark datasets, and weak reproducibility as persistent limitations of the field (Atlam et al., 2024).

This study aims to design a self-hosted blockchain-forensics system implemented in Rust and to provide a preliminary evaluation of its performance, security, and explainability using a practical prototype. Self-hosted means that the system is deployed within infrastructure controlled by the researcher or organisation. The object of study is a stream of confirmed Solana mainnet-beta transactions; the subject comprises normalization, graph representation, payment matching, and detection of suspicious patterns. The working hypothesis is that a Solana-specific representation combined with a Rust implementation can support a scalable and auditable analytical pipeline, although numerical advantages must be established by a reproducible experiment rather than an operational report alone.

Four objectives are addressed: (1) map blockchain-forensics methods to the Solana data model; (2) formalize a pipeline from read-only RPC to an Evidence Bundle; (3) specify explainable rules for wash trading, dusting, sandwich/MEV, and pump-and-dump; and (4) assess the reported Python and Rust measurements while accounting for internal consistency and threats to validity. The novelty is system-integrative: individually established techniques are adapted to Solana and combined in one auditable pipeline that links the original RPC response, normalized graph, heuristic version, and Evidence Bundle. The paper does not claim priority for BFS, threshold rules, or Rust language mechanisms individually.

RELATED WORK

Early Bitcoin studies demonstrated that address pseudonymity is compatible with recovery of substantial economic structure. Multi-input heuristics, change-address identification, and clustering can group addresses into probable entities, after which on-chain structure may be compared with external labels. A sound forensic conclusion must distinguish observation from interpretation: graph proximity or participation in one transaction may increase the probability of a relationship but does not itself prove common control (Meiklejohn et al., 2016; Conti et al., 2018).

For Ethereum, the analytical unit shifted from a simple transfer to a sequence of smart-contract interactions. Studies of Ponzi schemes use transaction features, bytecode, and contract behaviour (Chen et al., 2018; Bartoletti et al., 2020). DeFi attack analysis supplements transfer graphs with exchanges, borrowing, liquidation, and price-manipulation events (Zhou et al., 2023). These methods are relevant to Solana, but EVM-specific logs must be replaced by analysis of programs, account keys, inner instructions, and token-balance changes.

A separate body of work examines market manipulation. Pump-and-dump behaviour is identified from abnormal combinations of volume, returns, short-term price growth, and coordinated promotion; published studies stress the importance of temporal windows and careful threshold selection (Kamps & Kleinberg, 2018; Victor & Hagemann, 2019; Xu & Livshits, 2019; Hamrick et al., 2021). Wash trading appears as repeated cycles among related addresses

and artificial volume, but the same graph structure may arise from a market maker or aggregator. Detection should therefore account for economic outcome, repetition, asset, and contract role (Victor & Weintraud, 2021).

MEV research shows that transaction ordering and price changes can reveal front-run/victim/back-run structures in decentralised exchanges (Daian et al., 2020; Qin et al., 2022). Mechanical transfer of Ethereum rules to Solana is limited by the absence of an identical public mempool model, the use of bundle infrastructure, and slot-based execution. A Solana detector should therefore use the confirmed order and instruction set within a slot, the trading pair, swap direction, slippage, and initiator profit rather than signature adjacency alone.

AML/CFT analytics combines directed graphs, temporal and volume features, centrality, distance to known high-risk clusters, and machine learning. Explainable models matter because classes are imbalanced and false positives are costly. Pocher et al. (2023) demonstrate the applicability of machine learning to anomalous cryptocurrency transactions, while Hasan et al. (2024) emphasise explainability. The prototype studied here follows a conservative stage: deterministic and graph rules provide the rationale for a signal, while machine learning is treated as a later extension after a labelled benchmark dataset is available.

Solana-specific research is becoming a distinct strand. Transaction-network analysis reveals structural properties that differ from other platforms (Alizadeh & Khabbazian, 2025). Smart-contract security research demonstrates automated vulnerability detection for Solana programs (Cui et al., 2022), while analysis of failed transactions shows that they are an independent information source and should not be discarded before their causes are examined (Zheng et al., 2025). These findings support a multi-level evidence model that includes completed transfers and execution context.

Table 1. Synthesis of research directions and transferability to Solana

Direction	Observed data	Use	Transfer limitation
Bitcoin forensics	UTXO, addresses, amounts, time	Clustering and flow tracing	Solana uses an account model and programs
Ethereum/DeFi	Contract events, calls, DEX routes	Ponzi schemes, DeFi attacks, MEV	No direct equivalent of EVM logs
AML/CFT and ML	Graphs, temporal and volume features	Prioritisation of suspicious entities	Requires labels, calibration, and explainability
Solana network	Slots, account keys, instructions, SPL balances	Architecturally valid features	Few open benchmark datasets
Rust safety	Types, ownership, dependencies	Memory safety and parallelism	Does not exclude logic or supply-chain defects

Rust is selected for more than execution speed. Its ownership model and type system reduce memory-management errors and data races, but do not eliminate business-logic defects, unsafe code, or software-supply-chain risks. RustBelt provides formal foundations for Rust safety (Jung et al., 2018), while practical deployment still requires pinned dependencies, audits, and

invariant testing. Rust is therefore treated as one protective layer alongside read-only privileges, input validation, idempotency, and atomic state persistence.

MATERIALS AND METHODS

The study uses a combined design. A thematic review establishes scientific requirements, and the supplied practical report provides the prototype architecture and primary measurements. This design allows engineering choices to be assessed against the literature without conflating published findings with operational claims. The review covers work published from 2013 to 2026 on transaction-graph analysis, AML/CFT, Ponzi schemes, wash trading, pump-and-dump, MEV, Solana security, and Rust. Peer-reviewed articles, IEEE/ACM proceedings, and official protocol documentation were included; promotional comparisons and figures lacking a method were not treated as independent evidence.

The practical dataset is described as 10,000 Solana mainnet-beta transactions from slots 250,000,000–250,001,000. Python 3.12 and Rust 1.78 compiled with --release were compared on an AMD EPYC 7532 platform with 32 cores, 64 GB RAM, and NVMe storage. The measured outcomes were parsing time, graph construction with a bounded BFS, peak resident memory, and thread count. The source does not specify the extraction date, RPC provider, distribution of legacy and versioned transactions, success/failure ratio, number of repetitions, cache warm-up, or profiling instrument. These omissions are treated as limitations. The inconsistent per-transaction values in the source are regarded as a typographical error and are explicitly recalculated as $t = T/N$ from the aggregate measurements.

The unit of analysis is a confirmed transaction signature together with execution metadata. Each record should retain slot, block time, success/error status, account_keys, outer and inner instructions, log messages, SOL and SPL pre/post balances, mint, decimals, token-account owner, and program id. To preserve traceability, the raw RPC response is hashed and linked to the Parser version and retrieval time. A normalized record supplements rather than replaces the raw response.

Table 2. Experimental configuration reported in the practical study

Element	Reported value	Replication assessment
Data	10,000 mainnet-beta transactions; slots 250,000,000–250,001,000	Range specified; signatures and export unavailable
Python	3.12; single-threaded processing	Source code and package list unavailable
Rust	1.78; --release; Rayon, Tokio, serde	Cargo.lock and commit hash unavailable
Hardware	AMD EPYC 7532, 32 cores; 64 GB; NVMe	OS, frequencies, and background load unspecified
Metrics	Parsing, BFS depth=5, peak RSS, CPU threads	Repetitions and confidence intervals unspecified

The architecture comprises seven functional modules (Figure 1). RPC Reader performs read-only operations and stores no private-key material. SOL/SPL Parser resolves account_keys

indices and processes outer instructions, inner instructions, and balance changes. Data Normalizer converts amounts to integer atomic units and links each token account explicitly to its mint and owner. Temporal Multigraph represents an oriented temporal multigraph in which two vertices may be linked by multiple edges for different assets and programs. Detection Module calculates graph and temporal features. Evidence Bundle and Audit Log preserve facts, rule version, and the path that produced a signal; Analyst Notification delivers the result to an analyst.

Self-hosted deployment reduces dependence on a closed scoring platform and permits rules to be rerun against retained data. It also transfers responsibility for RPC credentials, dependency updates, backups, and integrity monitoring to the system owner. Self-hosting is therefore not treated as an automatic guarantee of privacy or security, but as a governance and control model whose effectiveness depends on operations.

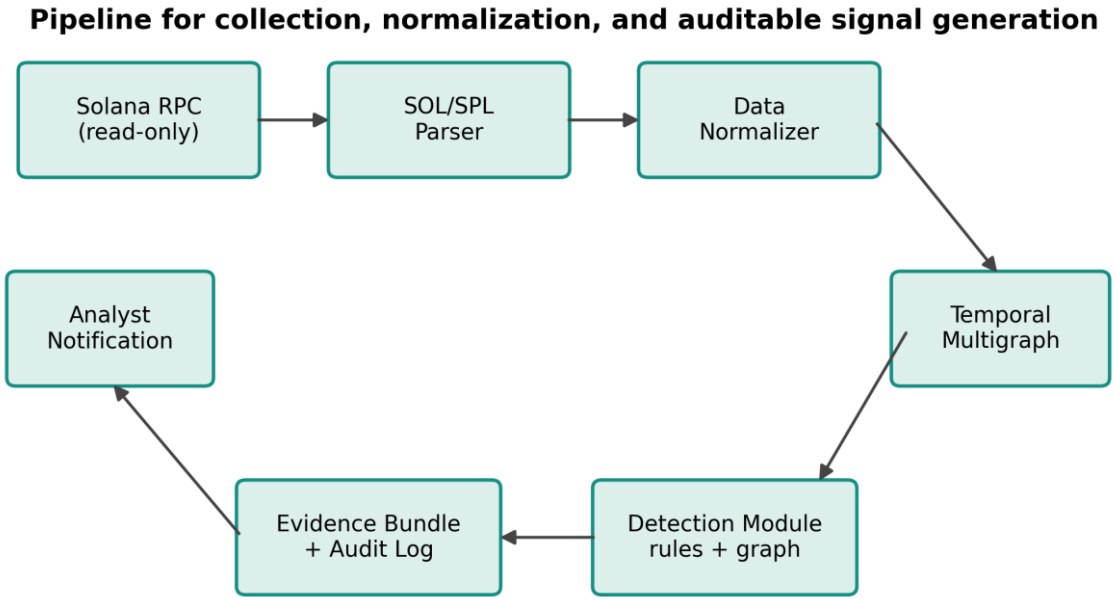


Figure 1. Architecture of the self-hosted blockchain-forensics system for Solana

A central error in SPL-transfer analysis is to equate a token account with a user's wallet. The normalized schema preserves at least four distinct entity types: wallet/owner, token account, mint, and program. A transfer is represented as an edge between owners only after explicit owner resolution; the original edge between token accounts is retained in parallel. This approach permits later verification when resolution rules change and prevents artificial fragmentation of one owner into several graph vertices.

Amounts are stored as unsigned integers in minimal units, while decimals are used only for display. This eliminates binary floating-point errors. For an expected payment, the prototype uses the relative deviation between the observed and target amount. The matching condition is defined as follows:

$$match = |txraw - targetraw| / max(targetraw, 1) \leq \tau$$

where txraw is the observed amount in atomic units, targetraw is the expected amount, and τ is the permitted relative deviation; the prototype sets τ = 0.02. A zero amount is rejected, and a processed signature must be marked idempotently. An SPL transfer fee is normally paid separately in SOL and does not reduce the token amount; therefore, a universal ±2% tolerance cannot be justified by network fees. It increases the risk of collision among similar expected

payments and should be calibrated by asset, business process, and concurrency. A stricter alternative is exact matching of integer amount_raw together with mint, destination owner, time window, and a unique order identifier.

Table 3. Core entities in the normalized Solana model

Entity	Identifier	Analytical role	Typical error
Wallet / owner	Public key	Assumed controlling entity	Treating every account as a wallet
Token account	Account public key	Balance holder for a specific mint	Failing to resolve the owner
Mint	Mint public key	Defines asset and decimals	Merging tokens with similar names
Program	Program id	Defines instruction semantics	Interpreting bytes without program id
Signature/slot	Signature + slot	Idempotency and temporal order	Processing one operation repeatedly

The graph is defined as G=(V,E). Vertices V include owners, token accounts, mints, and programs; every edge e∈E stores signature, slot, instruction index, asset, amount_raw, status, and action type. A multigraph is required because the same pair of participants may exchange several assets and invoke several programs within one time window. Time is not reduced to a block number: instruction order within a transaction and slot is relevant to MEV, while inter-slot windows are required for cycle and pump-and-dump analysis.

The wash-trading module performs a bounded cycle search to depth five, but creates a signal only when additional conditions are met: structural repetition, comparable returned volume, a short temporal window, and absence of a known service role. The dusting rule combines a low-value threshold, mass distribution, and an association between the sender or memo/URI and a risk indicator. A single micro-transfer without those conditions remains an observation rather than proof of an attack.

The sandwich/MEV rule analyses swaps of one token pair within a slot and checks direction, price change, the position of the potential victim, and a positive result for the closing operation. The non-standard term sandwAitch in the practical report is normalized to sandwich. The pump-and-dump module uses a sliding window for price and volume. The prototype's threshold of more than 500% price growth is treated as a configurable heuristic because published work shows that suitable parameters depend on liquidity and observation frequency (Kamps & Kleinberg, 2018; Xu & Livshits, 2019).

Every analytical signal is accompanied by an Evidence Bundle containing source signatures, slot range, normalized edges, feature values, triggered conditions, configuration version, a hash of the raw RPC response, and alternative explanations. This structure allows the conclusion to be repeated, the rule to be recalculated after a threshold change, and the signal to be reviewed by an expert. Without it, a signal is an operational notification rather than a

reproducible scientific result.

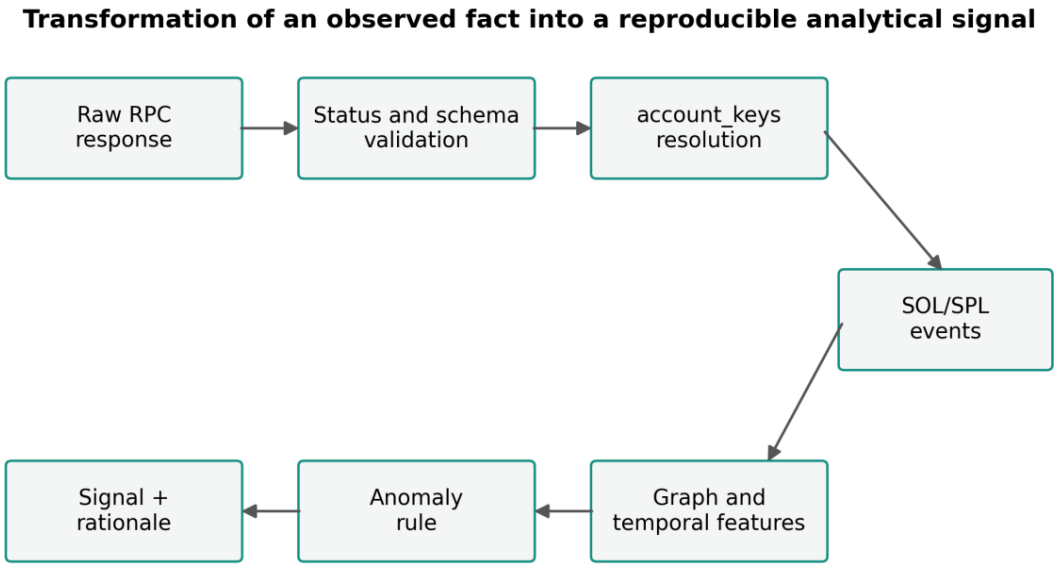


Figure 2. Sequence for generating an explainable analytical signal

Table 4. Explainable rules for suspicious-pattern detection

Pattern	Minimum features	Method	Required expert review
Wash trading	Cycle, repetition, comparable volumes, short window	BFS/DFS to depth 5 + filters	Service wallets, market making, aggregator
Dusting	Small amount, mass distribution, risk indicator	Threshold + sender cluster	Asset value and purpose of memo/NFT
Sandwich / MEV	Same pair, ordering, price, and profit	Instruction and slot analysis	Bundle/validator context and DEX route
Pump-and-dump	Price/volume spike, concentration, reversal	Sliding window + thresholds	Liquidity, news, and listing events

The security-by-design model includes least privilege, no private-key storage, TLS for RPC, strict deserialization, response-size limits, timeouts, rate limiting, signature-based idempotency, and atomic state persistence using write-temp–fsync–rename. An inter-process lock protects a state file but does not replace transactional storage in a multi-node deployment. A repeated run must safely recognise a known signature, and an RPC failure must not turn an incomplete response into a negative finding.

Performance is assessed using aggregate parsing time for 10,000 transactions, graph-construction time, and peak RSS. The ratio TPython/TRust is calculated for corresponding aggregate measurements, and average transaction time is $t = T/N$. This yields 0.38 ms for Python and 0.0038 ms for Rust, a ratio of 100. The ratio characterises two specific software configurations

rather than an isolated language effect. Precision, recall, and F1 are not calculated because the material contains no labelled control dataset with ground-truth classes.

Operational counts of processed transactions, signals, and incidents are analysed descriptively. Zero registered false matches is not equivalent to precision=1 without independent review of all positives and a search for missed cases. Similarly, the absence of known incidents does not prove system security; it records only the state of the monitoring log for the reported period.

Table 5. Threat model and design countermeasures

Threat	Vector	Countermeasure	Residual risk
RPC manipulation	False or incomplete response	TLS, multiple sources, hash raw response	Provider or endpoint compromise
Duplicate credit	Two workers process one signature	Unique signature, lock, idempotency	Distributed-coordination failure
Overflow	Incorrect amount arithmetic	u64/u128, checked arithmetic, boundary tests	Incorrect decimals conversion
Denial of service	Large response or frequent polling	Limits, timeout, backoff, queues	RPC quota exhaustion
Software supply chain	Compromised crate	Cargo.lock, cargo-audit, SBOM, pinning	Undetected dependency vulnerability

RESULTS AND DISCUSSION

The practical report gives 3,800 ms for Python and 38 ms for Rust when parsing 10,000 transactions; graph construction required 2,100 and 21 ms, respectively. Both aggregate pairs yield a ratio of 100:1. Peak memory is reported as 480 MB for Python and 42 MB for Rust, an approximately 11.4-fold reduction. The Rust implementation used 16 Rayon threads, whereas Python was described as single-threaded. The 100-fold difference therefore validly characterises the reported implementation-level comparison but does not isolate language, algorithm, data format, parallelism, or other engineering factors.

Average parsing time is calculated from the aggregate observations: 3,800/10,000 = 0.38 ms per transaction for Python and 38/10,000 = 0.0038 ms for Rust. Inconsistent per-transaction values in the source are interpreted as a typographical error and replaced by these calculated quantities. The ratio of the averages, like that of the totals, is 100:1. Because repeated runs and confidence intervals are unavailable, the result is a pilot estimate of an observed difference rather than a universal performance guarantee across workloads and hardware platforms.

As of May 2025, the practical material reports more than 1.2 million transactions processed across five networks with a 15-second polling interval. Mean RPC response time is reported as 85 ms for Solana and 120 ms for Ethereum. The log contains 47 wash-trading cluster signals and 12 dusting signals; registered false payment matches and known security incidents are both reported as zero. These values describe one deployment history but cannot serve directly as diagnostic-accuracy measures because denominators, independent labels, and a procedure for

identifying false negatives are unavailable.

Table 6. Performance results and internal-consistency check

Metric	Python	Rust	Ratio	Status
Parsing 10,000 transactions	3,800 ms	38 ms	100×	Aggregate measurements
Parsing 1 transaction	0.38 ms	0.0038 ms	100×	Calculated as T/N
BFS graph	2,100 ms	21 ms	100×	Aggregate measurements
Peak RSS	480 MB	42 MB	11.4×	Common measurement tool unspecified
CPU threads	1	16	16×	Language and parallelism differ

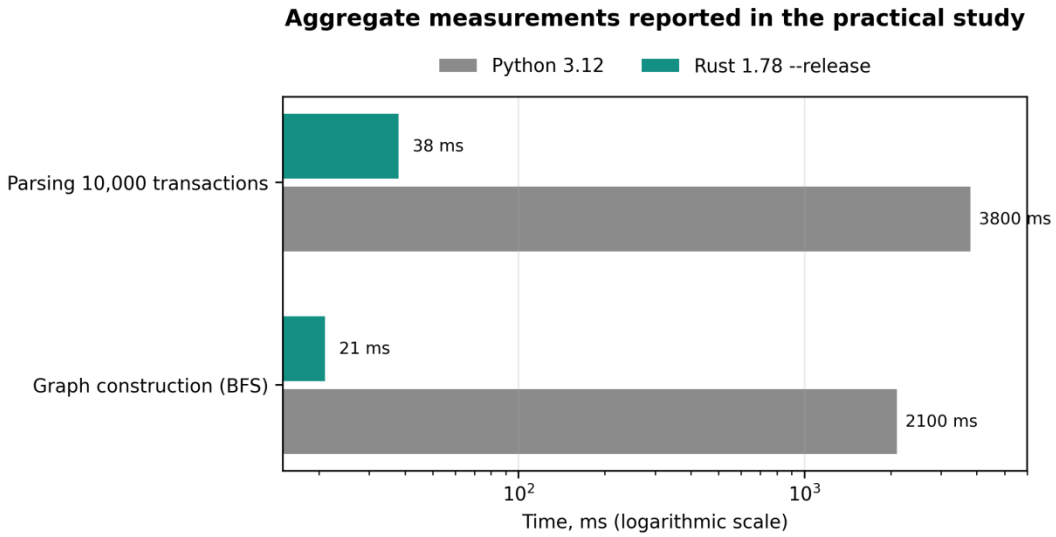


Figure 3. Comparison of aggregate Python and Rust execution times

A case dated 24 March 2025 concerns a transfer of 0.0001 SOL from an address previously flagged as phishing-related to a user address holding USDC. The system generated a dusting warning. Scientifically, the case demonstrates propagation through the pipeline 'micro-transfer + risk indicator → analytical signal'. It does not establish that a loss was prevented without evidence of the user's subsequent actions. The defensible result is timely generation of an explainable warning, not proven prevention of harm.

Deterministic payment matching also requires cautious interpretation. Zero known false matches may reflect low competition among expected payments, narrow time windows, or incomplete review. Quality measurement requires a log of all expected payments and on-chain candidates, followed by expert labelling of TP, FP, and FN. Only then can precision, recall, and F1 be calculated and exact amount_raw matching compared with a ±2% tolerance.

Table 7. Operational indicators and defensible interpretation

Indicator	Reported value	What it supports	What it does not support
Transactions processed	>1.2 million	Scale of one deployment	Complete network coverage
Wash-trading signals	47 clusters	Execution of the rule and log	47 proven fraud cases
Dusting signals	12	Detection of the configured pattern	Recall or absence of misses
False matches	0 in the log	No known false positives	Precision=1 without independent labels
Security incidents	0 in the log	No known incidents	Complete system security

The architecture supports a common conclusion in the literature: effective blockchain forensics depends on graph structure, temporal context, and reproducible transformations rather than a single risk score (Atlam et al., 2024). The contribution is limited to system-level adaptation for Solana through explicit retention of entity and instruction layers and a traceable link from the output to source facts. Immediate reduction to wallet-to-wallet transfers would be convenient but would destroy part of the evidence context and impede reinterpretation.

Graph cycles are consistent with established approaches to wash-trading detection, but prior work warns that their meaning is context-dependent (Victor & Weintraud, 2021). The bounded BFS should therefore be treated as a candidate generator. Temporal windows, economic outcome, and service labels can remove obvious false positives, while final specificity must be evaluated on labelled Solana DEX data.

Pump-and-dump rules use features reported for centralised and decentralised markets: abrupt volume, price increase, participant concentration, and reversal (Kamps & Kleinberg, 2018; Hamrick et al., 2021). A 500% threshold is convenient for a prototype but is not universal. One trade may produce that movement in an illiquid token, whereas manipulation in a liquid asset may use a smaller amplitude. Thresholds should consequently be normalised against historical volatility and market depth.

For MEV, an observed sequence must be distinguished from an inferred strategy. Ethereum sandwich behaviour is commonly related to ordering in the input stream and block; Solana additionally requires slot, instruction, DEX, and bundle context. The prototype starts with one-pair sequencing, but an A–V–B condition is insufficient. A future version should calculate pool reserve changes, post-fee profit, and links between the addresses responsible for the front and back operations (Daian et al., 2020; Qin et al., 2022).

The Rust findings are conceptually consistent with the use of a compiled language for intensive deserialization and parallel processing. Rust ownership does not automatically make an application secure: Parser may resolve an owner incorrectly, a rule may use an unsuitable threshold, and a dependency may be vulnerable. The practical security effect should be evaluated

with property-based tests, fuzzing, cargo-audit, SBOM, and invariant tests (Jung et al., 2018; Cui et al., 2022).

The practical value of self-hosting is control over data and explanatory logic. Unlike a closed service, the researcher can retain a raw RPC response, repeat parsing, and identify the exact rule version. Cost and feature comparisons with commercial products in the practical report are excluded from the evidentiary argument: prices and capabilities change, and no solution can be declared functionally superior without a standardised task.

allocation, and data-structure construction. The arithmetically consistent 100:1 ratio confirms a difference between the two measured configurations, but cannot be interpreted as a statistical estimate of the isolated Rust effect without repetitions and thread-count alignment.

Construct validity is limited because counts of analytical signals are not counts of confirmed attacks. Wash-trading and dusting results lack independent expert labels; '0 false positives' records no known errors rather than systematic validation. Error costs, inter-rater agreement, and ground-truth criteria are not defined.

External validity is restricted to one slot range and one hardware platform. Transaction-type distribution, congestion, versioned transactions, address lookup tables, and failed transactions may differ in another period. Operations covered five networks, but the paper concerns Solana; transfer to TRON, TON, Ethereum, or Litecoin is not evaluated.

Reproducibility requires publication of an anonymised signature list or resampling procedure, JSON schema, code, Cargo.lock, Python requirements, RPC configuration, benchmark scenario, and raw outputs from every run. Until such a package is available, the measurements should be treated as a pilot study. That status does not negate engineering utility, but limits statistical claims.

The ethical boundary is that public on-chain data does not eliminate the risk of erroneous attribution. An address or cluster should not be labelled criminal solely because an automated rule fired. The output is intended to prioritise investigation; attribution requires additional evidence, a legal basis, and a review or appeal procedure.

Recommended replication protocol. The following protocol is proposed to convert the pilot evaluation into a reproducible study:

1. Fix at least three non-overlapping slot ranges; retain signatures, successful/failed proportions, and transaction versions; publish the sampling script.
2. Record the commit hash of both implementations, Cargo.lock, requirements, operating system, CPU settings, thread counts, and RPC configuration; separate I/O and CPU-only modes.
3. Run at least 30 repetitions after controlled warm-up; randomise execution order; report the median, IQR, mean, and 95% confidence interval.
4. Compare single-threaded Rust with single-threaded Python, then evaluate Rust scaling at 1, 2, 4, 8, and 16 threads.
5. Create a labelled normal/wash/dusting/MEV/pump-and-dump corpus; use at least two experts and report agreement, confusion matrix, precision, recall, F1, and PR-AUC.
6. Conduct an ablation study without owner resolution, inner instructions, temporal context, and with alternative thresholds τ; measure the contribution of every component.

This protocol would address the principal threats to validity and test both the computational hypothesis and detection utility. A separate I/O benchmark would quantify RPC dependence, a CPU-only benchmark would measure parsing and graph-construction cost, and a labelled corpus would establish the quality of the analytical signals.

CONCLUSION

This paper combines a blockchain-forensics review with practical material on a self-hosted system implemented in Rust. Solana requires a platform-specific feature model that distinguishes wallet, token account, mint, and program; retains outer and inner instructions; and records execution status and slot context. The proposed pipeline is read-only RPC → schema validation → atomic-unit normalization → temporal multigraph → explainable rule → Evidence Bundle.

The pilot results show lower aggregate execution time and peak memory for the Rust implementation than for Python. After correction of a typographical error, the averages are 0.38 and 0.0038 ms per transaction, giving an observed ratio of 100:1. This result is valid for the reported configurations; however, the absence of repetitions and comparison of single-threaded Python with 16-threaded Rust prevent attribution of the entire difference to the programming language or transfer of the coefficient to other conditions without a controlled benchmark.

Rules for wash trading, dusting, sandwich/MEV, and pump-and-dump generate candidates for review rather than establish fraud. The scientific and practical value of the system lies in retaining observations, rule version, and alternative explanations. This approach makes a signal reproducible and reduces the risk of turning a probabilistic relationship into unjustified attribution.

Future work should provide an open Solana benchmark dataset, repeated measurements, ablation studies, expert labels, and formal parser invariants. Graph neural networks or other machine-learning models may be added after reliable labelling, but their outputs should also be accompanied by interpretable features and an Evidence Bundle.

CONFLICT OF INTEREST: The authors declare no conflict of interest.

FUNDING: This research received no external funding.

INSTITUTIONAL REVIEW BOARD STATEMENT (IRB): Ethical approval was not required because the study involved neither humans nor animals and used only public blockchain records.

INFORMED CONSENT: Not applicable.

DATA AVAILABILITY: Transaction signatures and source code required for full benchmark replication were not included in the supplied materials. Aggregate experimental parameters are reported in the article; publication of a replication package is recommended before final submission.

ACKNOWLEDGEMENTS: The authors acknowledge researchers and developers of open blockchain tools whose work informed the methodology.

DECLARATION OF GENERATIVE AI USE: Artificial-intelligence tools were used for manuscript structuring. The authors verified and approved the scientific claims, numerical data, references, and conclusions and retain full responsibility for the article.

REFERENCES

Alizadeh, S., & Khabbazian, M. (2025). Solana's transaction network: Analysis, insights, and comparison. EPJ Data Science, 14, Article 48. https://doi.org/10.1140/epjds/s13688-025-00561-x

Atlam, H. F., Ekuri, N., Azad, M. A., & Lallie, H. S. (2024). Blockchain forensics: A systematic literature review of techniques, applications, challenges, and future directions. Electronics, 13(17), 3568. https://doi.org/10.3390/electronics13173568

Bartoletti, M., Carta, S., Cimoli, T., & Saia, R. (2020). Dissecting Ponzi schemes on Ethereum: Identification, analysis, and impact. Future Generation Computer Systems, 102, 259–277. https://doi.org/10.1016/j.future.2019.08.014

Chen, W., Zheng, Z., Cui, J., Ngai, E. C. H., Zheng, P., & Zhou, Y. (2018). Detecting Ponzi schemes on Ethereum: Towards healthier blockchain technology. In Companion Proceedings of the Web Conference 2018 (pp. 1409–1418). ACM. https://doi.org/10.1145/3178876.3186046

Conti, M., Kumar, E. S., Lal, C., & Ruj, S. (2018). A survey on security and privacy issues of Bitcoin. IEEE Communications Surveys & Tutorials, 20(4), 3416–3452. https://doi.org/10.1109/COMST.2018.2842460

Cui, S., Zhao, G., Gao, Y., Tavu, T., & Huang, J. (2022). VRust: Automated vulnerability detection for Solana smart contracts. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (pp. 639–652). ACM. https://doi.org/10.1145/3548606.3560552

Daian, P., Goldfeder, S., Kell, T., Li, Y., Zhao, X., Bentov, I., Breidenbach, L., & Juels, A. (2020). Flash Boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In 2020 IEEE Symposium on Security and Privacy (pp. 910–927). IEEE. https://doi.org/10.1109/SP40000.2020.00040

Hamrick, J. T., Rouhi, F., Mukherjee, A., Feder, A., Gandal, N., Moore, T., & Vasek, M. (2021). An examination of the cryptocurrency pump-and-dump ecosystem. Information Processing & Management, 58(4), 102506. https://doi.org/10.1016/j.ipm.2021.102506

Hasan, M., Rahman, M. S., Janicke, H., & Sarker, I. H. (2024). Detecting anomalies in blockchain transactions using machine learning classifiers and explainability analysis. Blockchain: Research and Applications, 5, 100207. https://doi.org/10.1016/j.bcra.2024.100207

Jung, R., Jourdan, J.-H., Krebbers, R., & Dreyer, D. (2018). RustBelt: Securing the foundations of the Rust programming language. Proceedings of the ACM on Programming Languages, 2(POPL), Article 66. https://doi.org/10.1145/3158154

Kamps, J., & Kleinberg, B. (2018). To the moon: Defining and detecting cryptocurrency pump-and-dumps. Crime Science, 7, Article 18. https://doi.org/10.1186/s40163-018-0093-5

Li, X., Jiang, P., Chen, T., Luo, X., & Wen, Q. (2020). A survey on the security of blockchain systems. Future Generation Computer Systems, 107, 841–853. https://doi.org/10.1016/j.future.2017.08.020

Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G. M., & Savage, S. (2016). A fistful of Bitcoins: Characterizing payments among men with no names. Communications of the ACM, 59(4), 80–87. https://doi.org/10.1145/2896384

Pocher, N., Zichichi, M., Merizzi, F., Shafiq, M. Z., & Ferretti, S. (2023). Detecting anomalous cryptocurrency transactions: An AML/CFT application of machine learning-based forensics. Electronic Markets, 33, Article 37. https://doi.org/10.1007/s12525-023-00654-3

Qin, K., Zhou, L., & Gervais, A. (2022). Quantifying blockchain extractable value: How dark is the forest? In 2022 IEEE Symposium on Security and Privacy. IEEE. https://doi.org/10.1109/SP46214.2022.9833734

Solana Foundation. (2019). Sealevel: Parallel processing thousands of smart contracts. https://solana.com/news/sealevel---parallel-processing-thousands-of-smart-contracts

Solana Foundation. (2026). Tokens on Solana. https://solana.com/docs/tokens

Victor, F., & Hagemann, T. (2019). Cryptocurrency pump and dump schemes: Quantification and detection. In 2019 IEEE International Conference on Data Mining Workshops (pp. 244–251). IEEE. https://doi.org/10.1109/ICDMW.2019.00045

Victor, F., & Weintraud, A. M. (2021). Detecting and quantifying wash trading on decentralized cryptocurrency exchanges. In Proceedings of the Web Conference 2021 (pp. 23–32). ACM. https://doi.org/10.1145/3442381.3449824

Werner, S. M., Perez, D., Gudgeon, L., Klages-Mundt, A., Harz, D., & Knottenbelt, W. J. (2022). SoK: Decentralized finance (DeFi). In Proceedings of the 4th ACM Conference on Advances in Financial Technologies (pp. 30–46). ACM. https://doi.org/10.1145/3558535.3559780

Xu, J., & Livshits, B. (2019). The anatomy of a cryptocurrency pump-and-dump scheme. In 28th USENIX Security Symposium (pp. 1609–1625). USENIX Association. https://www.usenix.org/conference/usenixsecurity19/presentation/xu-jiahua

Yakovenko, A. (2017). Solana: A new architecture for a high performance blockchain (Version 0.8.13). https://solana.com/solana-whitepaper.pdf

Zheng, X., Wan, Z., Lo, D., Xie, D., & Yang, X. (2025). Why does my transaction fail? A first look at failed transactions on the Solana blockchain. arXiv. https://doi.org/10.48550/arXiv.2504.18055

Zhou, L., Xiong, X., Ernstberger, J., Chaliasos, S., Wang, Z., Wang, Y., Qin, K., Wattenhofer, R., Song, D., & Gervais, A. (2023). SoK: Decentralized finance (DeFi) attacks. In 2023 IEEE Symposium on Security and Privacy. IEEE. https://doi.org/10.1109/SP46215.2023.10179435

Авторлар туралы мәліметтер
Информация об авторах
Information about authors



Котлярова Ирина Александровна – «*Д.* Серікбаев атындағы Шығыс Қазақстан техникалық университеті» КеАҚ «Цифрлық технологиялар және жасанды интеллект мектебі» оқытушы, Өскемен қ., Қазақстан

Котлярова Ирина Александровна – преподаватель Школы цифровых технологий и искусственного интеллекта НАО «Восточно-Казахстанский технический университет им. *Д.* Серикбаева», г. Усть-Каменогорск, Казахстан

Kotlyarova Irina Aleksandrovna – NAO "D. Serikbaev East Kazakhstan Technical University" lecturer at Lecturer at the School of Digital Technologies and Artificial Intelligence of the NAO "D. Serikbayev East Kazakhstan Technical University", Ust-Kamenogorsk, Kazakhstan,
e-mail: IKotlyarova@edu.ektu.kz,
ORCID: https://orcid.org/0009-0006-5372-4207,



Попов Артур Андреевич – «*Д.* Серікбаев атындағы Шығыс Қазақстан техникалық университеті» КеАҚ «Цифрлық технологиялар және жасанды интеллект мектебі» студенті, Өскемен қ., Қазақстан

Попов Артур Андреевич – студент Школы цифровых технологий и искусственного интеллекта НАО «Восточно-Казахстанский технический университет им. *Д.* Серикбаева», г. Усть-Каменогорск, Казахстан

Popov Artur – student of the School of Digital Technologies and Artificial Intelligence of the NAO "D. Serikbayev East Kazakhstan Technical University", Ust-Kamenogorsk, Kazakhstan,
e-mail: p.artur952@gmail.com,
ORCID: https://orcid.org/0009-0005-8067-1772

	**Черных Тимур Равильевич** – «Д. Серікбаев атындағы Шығыс Қазақстан техникалық университеті» КеАҚ «Цифрлық технологиялар және жасанды интеллект мектебі» студенті, Өскемен қ., Қазақстан **Черных Тимур Равильевич** – студент Школы цифровых технологий и искусственного интеллекта НАО «Восточно-Казахстанский технический университет им. Д. Серикбаева», г. Усть-Каменогорск, Казахстан **Chernykh Timur** – student of the School of Digital Technologies and Artificial Intelligence of the NAO "D. Serikbayev East Kazakhstan Technical University", Ust-Kamenogorsk, Kazakhstan, e-mail: rewvtrew@gmail.com, ORCID: https://orcid.org/0009-0006-2645-3421
	**Киричёк Кирилл Николаевич** – «Д. Серікбаев атындағы Шығыс Қазақстан техникалық университеті» КеАҚ «Цифрлық технологиялар және жасанды интеллект мектебі» студенті, Өскемен қ., Қазақстан **Киричёк Кирилл Николаевич** – студент Школы цифровых технологий и искусственного интеллекта НАО «Восточно-Казахстанский технический университет им. Д. Серикбаева», г. Усть-Каменогорск, Казахстан **Kirichyok Kirill** – student of the School of Digital Technologies and Artificial Intelligence of the NAO "D. Serikbayev East Kazakhstan Technical University", Ust-Kamenogorsk, Kazakhstan, e-mail: kirichek331@gmail.com, ORCID: https://orcid.org/0009-0003-3095-3582
	**Петров Илья Александрович** – «Д. Серікбаев атындағы Шығыс Қазақстан техникалық университеті» КеАҚ «Цифрлық технологиялар және жасанды интеллект мектебі» студенті, Өскемен қ., Қазақстан **Петров Илья Александрович** – студент Школы цифровых технологий и искусственного интеллекта НАО «Восточно-Казахстанский технический университет им. Д. Серикбаева», г. Усть-Каменогорск, Казахстан **Petrov Ilya** – student of the School of Digital Technologies and Artificial Intelligence of the NAO "D. Serikbayev East Kazakhstan Technical University", Ust-Kamenogorsk, Kazakhstan, e-mail: petr.ilya142@gmail.com, ORCID: https://orcid.org/0009-0006-7946-2639