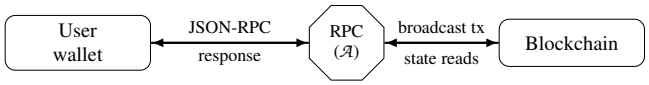
SoK: Blockchain RPCs – Centralized Gateways to a Decentralized Network

Sandhya Saravanan *UCSC*
Nihal Talur *UCSC*
Ioannis Demertzis *UCSC*
Dimitris Mouris *Snap Inc.*

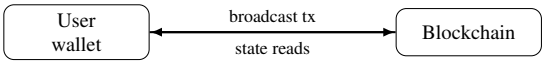
Abstract

Blockchains promise permissionless, decentralized access, yet nearly every interaction a user has with a blockchain first passes through a handful of *centralized* Remote Procedure Call (RPC) providers. About 70-80% of the RPC traffic flows through just Alchemy and Infura – invisible intermediaries with full visibility into personal information: billing details, transaction contents, wallet addresses, IP addresses, and behavioral patterns before requests even reach the chain. Existing privacy-preserving RPC designs are often evaluated under a single vague notion of privacy, although hiding identity, network metadata, and application-layer contents yield fundamentally different protections. We show this distinction has direct security consequences: while application-layer information remains visible, the RPC can carry out deanonymization, censorship, and maximal extractable value (MEV) attacks; and post deanonymization, selectively poison its responses.

Inspired by ecosystem transparency efforts such as L2BEAT and Walletbeat, we formalize client–RPC exposure through leakage profiles that capture both per-interaction disclosure and correlations across interactions, and use them to build a leakage-and-attack framework for RPC privacy. We apply this framework to systematize existing mitigation techniques, organizing them into leakage classes (Type 0–6) ordered by deployment difficulty, where Type 0 leaks everything in plaintext and Type 6 hides all of the request contents. These Types are not exhaustive of all possible system designs, but capture representative leakage regimes along the privacy–deployability spectrum and provide an extensible basis for classifying new architectures. This shows the attacks are by-design consequences of what each architecture exposes, not implementation bugs: all surveyed attacks survive every default and opt-in mitigation deployed today, and are closed only by a full-stack architecture that also protects application-layer contents. To ground these claims, we instantiate the attacks against each Type's leakage as proofs of concept on a dataset of client–RPC interaction logs that we create by replaying Ethereum methods such as `eth_sendRawTransaction` and `eth_getBalance` through the Ganache RPC endpoint.



(a) Reality: every action through a centralized RPC. $\mathcal{A}$ = adversary



(b) User mental model: direct interaction.

Figure 1: The RPC gap: what users believe vs reality.

Finally, we extend the framework to other blockchain intermediaries, giving a common language for reasoning about leakage across the stack to enable future research.

1 Introduction

The inverted decentralization promise. Decentralized systems such as blockchains were conceived to remove trusted intermediaries: to let users hold assets, transact, and run applications without a bank, platform, or any other central party sitting between them and their money or data [30, 153]. Nearly two decades on, this promise of permissionless, censorship-resistant access has been inverted at the very point it should hold. Virtually every wallet action, such as sending a transaction, checking a balance, or simulating a trade, is first routed through a small set of centralized Remote Procedure Call (RPC) providers such as Infura and Alchemy, invisible intermediaries that observe every request before it ever reaches the chain [139, 214]. The trust these systems removed from centralized institutions has reappeared one layer lower, at the point of access. Figure 1 illustrates the gap between how users think the system works and how it actually works.

The inevitable dependence on RPC providers. Reading from or writing to the chain requires a fully synchronized

node, too resource-intensive for most users to run. Even the minimum is demanding; an 8 GB RAM, 2 TB SSD, 10+ Mbit/s machine, and Ethereum's recommended specification is higher still: a fast multi-core CPU, 16 GB of RAM, 2+ TB of fast SSD storage, and 25+ Mbit/s of bandwidth, atop a live state of roughly 245 GiB (as of 2024) that grows continuously and must stay in sync [77, 185]. RPC providers absorb this cost: they run the full nodes and expose a hosted JSON-RPC endpoint so that wallets, decentralized applications (dApps), software development kits (SDKs), indexers, and bots can check balances, simulate and submit transactions through a single request [139]. This dependence places a few providers on the default path of nearly the entire ecosystem: 70–80% of RPC traffic flows through just Alchemy and Infura [23, 193, 216] for Ethereum and connected chains; MetaMask, with over 30 million monthly active users, ships Infura as its default endpoint, which most users never change; Infura alone serves more than 430,000 developers and over $1 trillion in annualized on-chain ETH transaction volume [54, 149]; Alchemy reports 100 million end users across more than 100 networks [4]; Ankr handles 6 billion requests per day [110].

Privacy compromise: RPCs see requests in the clear. These providers see every client request in the clear: sign-up information (e.g. name, email, billing details) under API-key access, and on the default wallet path, where sign-up is absent, the source IP address, the wallet addresses being queried, and the full contents of every read and write request. Despite years of work on private modes of RPC access, including Tor [70], mixnet relays [69, 165], encrypted mempools [20, 78], and stealth addresses [210–212], this default path remains fully exposed: existing mitigations are undeployed on Ethereum L1 [20, 72, 124, 137], discontinued [17, 106], opt-in with low adoption [69, 70, 165, 212], or shift trust to another single party [16, 83, 166]. Among the top 200 RPC services listed by CompareNodes [53], we found that the only deployed private RPC that hides information from the downstream RPC is the TEE-relay 1RPC [16], which conceals only the client's network identity and not application-layer contents, and self-reports 10M+ monthly users across 50+ chains; the mixnet-based HOPR RPCh [106] has been paused, and Kohaku [124], the most comprehensive attempt to close the gap, remains in development. Even deployed mitigations see little real-world use: stealth addresses, which hide a recipient behind a fresh one-time address per payment, cover roughly 0.002% of Ethereum transactions and 24K users across their two deployed implementations (Umbra and Fluidkey) [88, 212], while network-layer mitigations that hide the client's IP address (e.g., Tor, mixnets) add enough latency that, from usability studies of Tor and other real-time applications, degrade interactive wallet use [65, 196]. The only configuration that truly removes RPC leakage is to run one's own node and query it locally, which reintroduces precisely the hardware and operational burden that RPCs exist to eliminate. Accordingly, the threat model must consider what an RPC can do with the information it observes.

The RPC as adversary—From observation to attack. Unlike a passive blockchain observer that only sees confirmed transactions, the RPC sees and acts on every step before confirmation — balance checks, gas-price probing, contract simulations, failed attempts, submissions and receipt polling. This places several widely studied classes of blockchain-privacy and security attacks within reach of a single party. *(a) Deanonymization*: the RPC can reconstruct user portfolios more directly than the TRAP attack [214], which must correlate submission timestamps observed on-path with on-chain confirmation timestamps to link an IP to a wallet address. Once it has linked a request stream to a specific client this way, the RPC can go further and return forged responses tailored to that client, steering their exact next action into a disadvantageous one as in reported incidents where malicious RPC nodes fabricated wallet balances and forged chain state to trigger cross-chain execution against non-existent events [40, 108, 127, 186, 198]. *(b) Censorship*: the RPC can block by sanctioned address, geographical region, or prohibited contract before a transaction ever reaches the chain as already exercised when, within days of the U.S. Treasury's Office of Foreign Assets Control (OFAC) sanctioning Tornado Cash in 2022, Infura and Alchemy blocked requests to the mixer [60, 205], and MetaMask and Infura briefly cut off users in Venezuela [155]; Ethereum co-founder Buterin notes that many providers already exclude entire countries [32, 209]. Censorship is inherently selective; major RPC load balancers route by client IP, API key, or timing, enabling attacks on a specific dApp, client, or visit time [130], and time-critical: many DeFi protocols depend on emergency pauses, governance cancellations, and loan repayments [188] that a malicious RPC can delay past their deadlines. *(c) Sandwich MEV*: the RPC observes transaction details in the clear before submission; combined with control over the ordering of its own transactions relative to the victim's, this gives it everything it needs to front-run any targeted transaction. MEV-protect RPCs [114], adopted precisely to avoid front-running, concentrate signed transactions at a single receiving endpoint and so amplify rather than remove this vector. Turning that concentrated visibility into an executed sandwich still requires ordering control, but colluding with (or being) the block builder supplies exactly that, and is increasingly practical to arrange given how concentrated block building has become—private order flow is a documented centralizing force in PBS [18, 19, 98], and just three builders produced 80% of all MEV-Boost blocks over a recent six-month measurement period [162]. The RPC need not attack everyone: selective attacks on high-value traders or sanctioned addresses are harder to detect and easily mistaken for congestion, failed propagation, or ordinary provider error.

Real world stakes—From data to physical harm/crimes. An RPC that can link real-world identities to on-chain holdings exposes users to more than digital harm. Physical "wrench attacks" on known crypto holders reached 72 ver-

ified incidents in 2025, a 75% increase over 2024, totaling $40.9 million in confirmed losses, with kidnapping the most common method [38]. These attacks are targeted, not random: perpetrators select victims whose holdings are known to them [161], and public disclosure of holdings is a documented way attackers build target lists [39] – exactly the information an RPC holds. Recent cases include: a crypto investor tortured for more than two weeks in a New York kidnapping [144], Ledger co-founder David Balland abducted and mutilated for a multimillion-dollar Bitcoin ransom [51], and the father of a French crypto entrepreneur, whose finger was severed to extort ransom [48]. Even short of violence, an RPC that concentrates identities, balances, and behavioral histories for millions of users is a single point of compromise. In 2025, attackers bribed Coinbase support agents to access the data of 69,461 customers–names, contact information, government-ID images, balance snapshots, and transaction histories–and demanded a $20 million ransom [28, 96]. Providers are also compelled to hand over data: French authorities obtained Telegram user data after its founder's arrest [73, 128], and the VPN providers PureVPN and IPVanish turned over connection logs to U.S. authorities despite no-log claims [143, 195]. A 2024 industry survey reports that 83% of organizations experienced at least one insider attack [107]. Thus, an honest RPC, with its sheer access to data invites insider misuse, external compromise, and legal compulsion.

The gap—Prior work sees only fragments. Stakes this high call for a complete accounting of what the RPC sees and what that visibility enables, yet, no prior work that we surveyed provides one. Measurement studies document the concentration of RPC traffic and pervasive IP collection by RPCs [79, 130, 139], but stop at showing that leakage happens without quantifying what it enables. The documented censorship that followed the OFAC sanctions on Tornado Cash [209], is treated as an isolated incident rather than one consequence of a common leakage surface. Attack-side work such as TRAP [214] links a known IP to an on-chain address via timestamp matching, yet assumes only an on-path network adversary between client and provider — strictly weaker than the RPC itself, which terminates the Transport Layer Security (TLS) connection and gets every request in the clear for free. Together, these limitations motivate our central question:

What exactly does an RPC see, which attacks does each piece of that view enable, and how to close them?

Questions blockchain users would demand of any bank–yet never ask of the unknown intermediaries that replaced them.

Our Contribution/SoK. This paper answers all three parts of that question under a threat model that treats the RPC provider itself as the adversary: an active party that sees every request in the clear, since TLS terminates at its endpoint, and that can drop, delay, or inject requests, and route transactions to preferred downstream parties, rather than the weaker on path adversary of prior work [214]. An adversarial RPC also includes insider misuse and external compromise of an honest RPC. Under this model, we formalize what each RPC access architecture leaks (Contributions 1&2), connect this to attacks, inferring which architectures close which attacks (Contributions 3), and extend this formalism to other blockchain intermediaries (Contribution 4). We use Ethereum as our running example, but this work extends to other chains too.

Contribution 1—Leakage framework. To answer the first part of our question: what an RPC sees, observe that a wallet action is not a single blockchain operation but a sequence of JSON-RPC reads and writes: balance queries, gas estimation, transaction simulation, submission, and receipt polling. Even if each request is individually benign, their correlation reveals the user's address set, protocol interests, transaction intent, timing behavior, and sometimes real-world identity. These cross-request correlations motivate a leakage model defined over interaction sequences rather than isolated calls.

Leakage-profile abstractions that specify exactly what an observer/attacker learns are well-developed in searchable encryption [25, 67], secure multi-party computation [135], and anonymity [91], but this formal treatment has not been carried over to the blockchain access layer [42, 102]—leaving today's mitigations ad-hoc: stealth addresses ensure unlinkability of a transaction to the sender address but not the IP, encrypted mempools hide transaction parameters but not the sender, and no principled account exists of what leakage remains or which attacks a given deployment actually rules out.

We close this gap with a leakage framework (Section 3) that models each client–RPC exchange as an interaction tuple over sender identity, network metadata, application-layer content, timing, and volume. A leakage function projects each exchange onto what the RPC actually sees, and applying it across a sequence yields a *leakage profile*—a design-level account of what an architecture reveals when its cryptography works as specified. Appendix B instantiates this framework across three real-world customer segments, showing that the same leakage component carries different stakes depending on the requesting customer: repeated address reads expose a trading firm's strategy input set [45], and a compliance firm's active investigation targets [222]. By making explicit which components remain exposed, this shows exactly where mitigations must be composed.

Contribution 2—Systematization of RPC access architectures. We then apply the leakage framework to systematize deployed and proposed RPC access architectures in Section 3.1. For each architecture or mitigation primitive including API-key access, wallet-mediated access, VPNs, Tor and mixnets, TEE relays, stealth addresses, encrypted mempools, PIR/ORAM reads, and secure-computation-based RPCs, we ask which leakage components remain visible to the RPC, which are suppressed, what trust assumption is introduced, and whether the design is deployed, opt-in, or research-stage. Because components can be suppressed in many combina-

tions (enumerating every subset would obscure the design tradeoffs that matter), we compose the primitives into a cumulative hierarchy of leakage classes (Type 0–6, Section 3.2), each suppressing one more component than the last:

- *Type 0:* exposes identity, network metadata, and application-layer contents in plaintext;
- *Type 1:* removes explicit account identity;
- *Type 2:* suppresses network identity;
- *Type 3:* hides link between write request and sender's long-term wallet address (transaction submissions);
- *Type 4:* suppresses remaining write request parameters;
- *Type 5:* suppresses the queried address of read requests;
- *Type 6:* suppresses transaction-simulation contents;

Type 6 leaves invoked method, timing, and volume visible by design, which a variant, Type 6* closes. Independent support for this design comes from outside blockchains: Apple's maturity ladder for its Private Cloud Compute (PCC), presented at the Confidential Computing Summit 2026 [125], follows the same cumulative-suppression structure. The types are organized based on increasing deployment difficulty. Types 0 and 1 are the deployed defaults of every major provider and wallet, Types 2 and 3 are opt-in with low adoption, and Types 4, 5 and 6 are research-stage with no production wallet deployment on Ethereum L1. The taxonomy is thus based not on product labels such as "private RPC" or "MEV protection", but on the exact leakage profile each architecture leaves behind.

Contribution 3—Attack framework/Leakage-abuse analysis. Leakage-abuse attacks give leakage profiles their meaning (e.g., Searchable Encryption area [37, 111, 120]); a leakage function is only an abstraction until an attack shows what it actually reveals [25, 67]. We introduce an attack framework (Section 4) that turns the RPC's leakage profiles into concrete consequences: deanonymization, censorship, and sandwich MEV. Welfare-improving MEV such as arbitrage and liquidations read public post-execution state (pool reserves, health factors), whereas extractive MEV such as sandwiches need pre-execution visibility into the victim's pool, direction, amount, and slippage. This pre-execution visibility is the RPC's exact vantage, and combined with the greater need to eliminate extractive MEV over welfare-improving MEV, informs our focus on sandwiches. Since these attacks consume design-level leakage rather than bugs, misconfigurations, or weak parameters, a successful attack applies to every implementation in that leakage class. Each attack is a tuple over the adversary's capabilities, prior knowledge, algorithm, and goal, evaluated against a leakage profile. Suppressing a field, say the IP address via Tor can only shrink the attack surface, never enlarge it, so a profile pins down exactly which attacks remain possible and why isolated defenses fail. We instantiate[1] the attacks on an illustrative client-RPC interaction dataset we create by replaying Ethereum methods like `eth_sendRawTransaction` and `eth_getBalance` through the Ganache [202] RPC endpoint, grounded in real Ethereum activity and constructed to be realistic enough to meaningfully evaluate leakage-based attacks in Appendix D.

Contribution 4—From the RPC to the full stack. We focus on the Ethereum Layer-1 (L1) RPC, but this framework extends to other blockchain intermediaries. In Appendix A, we lift the RPC-specific view to a party-specific stack-wide view, where the adversary may be a wallet, RPC provider, P2P node, builder, relay, validator, sequencer and so on. We define a collusion-composed view that combines leakage observed by multiple intermediaries. This captures access-layer deanonymization, wallet telemetry leakage, relay/builder censorship, P2P deanonymization, sequencer censorship, and cross-layer correlation attacks.

Future Work. Read-address privacy (Type 5) is needed to degrade deanonymization; censorship and sandwich MEV survive until transaction-simulation contents are hidden (Type 6). The framework is extensible rather than exhaustive: leakage components are currently attack-focused, the network model uses only client IP, attack instantiations cover three classes, and evaluation uses constructed workflows not production traffic. Moreover, the framework is privacy-centric and analyzes an RPC largely in isolation. These boundaries motivate Section 5's directions on residual metadata, production validation, integrity, deployment practicality, and cross-layer leakage.

2 Preliminaries

Blockchain. A blockchain maintains a shared state mapping addresses to balances, nonces, code, and storage, plus an append-only ledger of the transactions that update it [153,220]. We use account-based Ethereum as our running example [30] (unlike UTXO chains [153]). Each address has a balance and monotonic nonce, which must be queried and incremented per submitted transaction. Portfolio reconstruction assumes this account model and extends to other account-based chains.

Consensus, Ordering, and Block Production. Ethereum uses proof-of-stake (PoS) [35]: time is divided into 12-second slots, each with one validator selected as block *proposer*. A submitted transaction first waits in the *mempool*, the set of unconfirmed transactions pending inclusion; gossiped to all peers if *public*, forwarded without gossip to a specific builder if *private*. In practice, block production follows out-of-protocol proposer–builder separation (PBS, e.g. MEV-Boost) [84, 99], with four roles: *searchers* construct profitable transaction sequences and submit them via `eth_sendBundle` as atomic *bundles*, included in order in full or not at all [81]; *builders* assemble candidate blocks from

[1] We release our dataset, attack implementations, and evaluation at https://anonymous.4open.science/r/Blockchain_Centralization_RPC-E355/README.md.

mempool and bundles, bidding for inclusion; *relays* forward the winning builder's block to the proposer; and the *proposer* selects the most profitable candidate.

Gas and fees. A transaction's fee is its gas usage (cost of its operations) times a per-unit price, which EIP-1559 [34] splits into a protocol-set burned *base fee* and a *priority fee* (tip) paid to the proposer. Builders order transactions by descending tip, so the fee influences inclusion and position.

Client–RPC Interface. End users interact through wallets, which manage key pairs, sign transactions, and query chain state via the pipeline *Wallet* $\rightarrow$ *RPC provider* $\rightarrow$ *Blockchain* using JSON-RPC 2.0 [117] over a TLS [173] session that *terminates at the RPC*, so TLS shields requests only from on-path observers while the provider sees every request and response in plaintext (Fig. 1). We separate RPC operations into *reads*, which query chain state without changing it (balance checks, contract simulations, gas estimates, receipt polling), and *writes*, which submit signed transactions (transfers, swaps, repayments, governance votes, emergency pauses); Section 3 formalizes what each exposes.

Wallet workflows. A user-facing action usually triggers an ordered sequence of JSON-RPC calls, a *workflow*, whose correlation reveals more than any single request. The two representative workflows below are grounded in standard wallet-client behavior; verified against open-source implementations of MetaMask, ethers.js, and viem [148]. A *portfolio refresh* queries each address's ETH balance via `eth_getBalance`, transaction count via `eth_getTransactionCount`, and ERC-20 token balances via `balanceOf` through `eth_call`, plus the current gas-price estimate via the parameterless `eth_gasPrice`. This reveals the user's address set, token interests, balances, and timing. An *ETH transfer* queries the account's transaction count via `eth_getTransactionCount` and recent fee history via `eth_feeHistory`; simulates gas consumption via `eth_estimateGas` and returned data via `eth_call`; submits via `eth_sendRawTransaction`; and polls via `eth_getTransactionReceipt`. Thus, the simulations expose the transaction to the RPC before submission.

Smart Contracts, DeFi, and AMMs. Smart contracts execute in the Ethereum Virtual Machine (EVM) [104, 220] and are invoked through ABI-encoded calldata [187] identifying the target function and arguments. Decentralized exchanges (DEXs) such as Uniswap [1] implement swaps through smart contracts that act as automated market makers (AMMs): a constant-product pool holds reserves x, y of a token pair with $x \cdot y = k$, so (ignoring fees) swapping Δx for Δy satisfies $(x+\Delta x)(y-\Delta y) = k$; users bound slippage by specifying a minimum output amountOutMin. Swap calldata thus reveals the input amount, token pair, and acceptable-price bound—the exact inputs for a sandwich attack (Section 4.4).

Lending, Health Factor, and Liquidation. In Aave [90], an over-collateralized position (against which borrower has drawn pooled deposit) becomes liquidatable when its health factor < 1, allowing a liquidator to repay debt and seize collateral plus a bonus. The pool contract's `getUserAccountData` returns the position's collateral, debt, and health factor.

Network-layer proxies (hiding network identity). A network-layer proxy is an intermediary between the client and RPC to hide the client's IP. Inspired by [136], we abstract it as a *secure proxy* that preserves request confidentiality and integrity. A *VPN* [71] tunnels TLS-encrypted traffic through a server that cannot inspect contents but sees the client's network identity, timing, and signup data. A *TEE relay* [16] decrypts traffic inside an attested enclave, strips identifying metadata, then re-issues the requests, with confidentiality and integrity rooted in TEE guarantees. *Oblivious HTTP* [197] splits trust: an on-path relay sees the client IP but not the encrypted contents, and the gateway (i.e. RPC) decrypts the contents only after the relay strips the IP. *Tor and mixnets* [44, 165, 199] chain intermediaries to decentralize trust; mixnets additionally reshape timing and volume at every hop (randomized delays, batch reordering, cover traffic), unlinking ingress from egress traffic. A *DC-net* [43] has each client contribute a fixed-size encrypted share per round (real or placeholder), that servers aggregate into an output revealing the message but not its sender; anonymity holds with one honest server, with integrity from additively homomorphic vector commitments [85, 177]. A *TEE oblivious mailbox* [91] buffers client messages in an enclave that the destination pulls on a pre-committed schedule, hiding submission timing.

Sender-address privacy. *Stealth addresses* [151, 212, 213] let a sender derive, via Elliptic-Curve Diffie–Hellman (ECDH) from the recipient's published meta-address, a fresh one-time address that the recipient locates with a viewing key and later spends from without directly exposing the recipient's long-term address as the sender; ERC-5564 [210] and ERC-6538 [211] standardize stealth address generation and meta-address registration. *Shielded pools* [21, 36, 150] hide internal pool transfers using zero-knowledge proofs.

Encrypted Mempools. Encrypted mempools hand the RPC a ciphertext that decrypts only after a configurable trigger (e.g. once ordering is fixed), instantiated three ways. *Threshold encryption* [2, 27, 78] encrypts under a t-of-n committee key, confidential unless t members collude. *VDF/timelock encryption* [3, 122] prevents decryption until a prescribed delay elapses via verifiable delay functions or a beacon releases the key, thus guaranteeing eventual decryption [72, 137]. *TEE-based encryption* [87, 157, 180] binds the key to an attested enclave that decrypts after the trigger and re-emits the plaintext over an authenticated channel.

Private reads and secure computation. *PIR/ORAM* [29, 47, 95, 103, 146, 170, 226] hide what a read reveals: Private Information Retrieval fetches a database entry without the server learning the index, and Oblivious RAM additionally hides cross-query access patterns; single-server PIR rests on cryptographic hardness, multi-server PIR additionally on non-collusion, while *TEE+ORAM* [141, 158] replaces the hardness assumption with an attested enclave reading ORAM-

protected storage—side-channel-resistant and orders of magnitude cheaper. *Secure computation* [57, 94, 135] extends this from fetching state to evaluating functions on private inputs (e.g., an encrypted simulation), revealing only the output; instantiations differ in trust model: MPC [135] distributes computation across non-colluding parties, FHE [94] computes on ciphertexts at a single server, and TEE-based execution [16, 57, 158] relies on hardware-attested isolation.

Deanonymization and Portfolio Reconstruction. The RPC can group wallet addresses by owner; attaching a real-world identity then exposes financial holdings, behavioral patterns, and business relationships. Prior deanonymization work targets the consensus P2P layer [100], the network P2P layer [131], or the public ledger—via clustering, change-address inference, and graph learning [68, 134, 145, 169, 172, 208, 222, 227]—none of which accounts for the RPC's complete visibility into pre-confirmation request contents.

Censorship. Once an RPC can link requests to a stable client i.e. deanonymize, it can censor specific users, jurisdictions, or categories of on-chain activity—a well-motivated access-layer threat [130, 209]. Transactions can also be blocked before inclusion via mempool denial-of-service [132, 217].

MEV and Sandwich Attacks. Transactions execute in order; so whoever controls ordering, inclusion, or exclusion can exploit this control to extract *maximal extractable value* (MEV) [61]. The standard taxonomy separates *welfare-improving* MEV (arbitrage, liquidations), which aligns prices and keeps lending protocols solvent, from *extractive* MEV (sandwiches, JIT liquidity, time-bandit reorgs) [61, 163, 171]. A *sandwich attack* places a front-run tx_F immediately before and a back-run tx_B after a victim swap tx_V: on the constant-product pool, tx_F pushes the price so tx_V executes at a degraded price, bounded only by its amountOutMin; and tx_B sells back into the displaced reserves, capturing the difference. The searcher submits the triple as a bundle, or, without bundle support, places tx_F and tx_B by descending priority fee in the public mempool. Two interventions are often conflated with closing this surface: PBS opens the builder role, changing *who* extracts MEV but not *what* is extractable, so sandwiches survive it [31]; encrypted mempools [2, 27, 78] hide an isolated submission's parameters until after ordering, collapsing the sandwich surface if absent in preceding plaintext simulations.

3 Our Formal Framework for RPC Leakage

Existing approaches to RPC privacy are ad-hoc combinations of privacy-enhancing technologies and while these are interesting exploratory efforts, they lack systematic methods to evaluate their actual privacy properties. To address this gap, we develop a formal framework for *design-level leakage*, the information a perfectly implemented RPC architecture reveals when its cryptographic protocols satisfy their stated guarantees, providing a common language for stating and comparing privacy guarantees across RPC architectures and linking concrete leakage components to concrete attacks (Section 4). Following the leakage-function methodology used in prior privacy and cryptographic literature, our framework is not an exhaustive enumeration of leakage components, privacy attacks or defenses. The concrete leakage components, architectures, and attacks studied below are illustrative instantiations, and new leakage-abuse attacks or defenses can be incorporated by specifying the leakage they consume or suppress without changing the framework.

First, we establish a precise model for RPC interactions that describes the exchange. We model each client–RPC request–response exchange as an interaction tuple.

Interaction Tuple. Let $\mathcal{F}$ denote the set of client-requested blockchain functionalities, such as `eth_getBalance`, `eth_call`, `eth_estimateGas`, `eth_sendRawTransaction`, `eth_getTransactionReceipt`, and `eth_getLogs`. An *interaction tuple* for functionality $F \in \mathcal{F}$ is one RPC request–response exchange $r_F = (x_{\mathsf{id}}, x_{\mathsf{net}}, x_{\mathsf{app}}, x_{\mathsf{time}}, x_{\mathsf{vol}})$.

Here x_{id} contains client identity information obtained during sign up or authentication: name, email, billing information and API key that binds later RPC requests to this identity, x_{net} contains network and transport metadata such as source IP, proxy/VPN/Tor exit identity, routing metadata, and connection metadata; x_{app} contains application-layer request/response contents; x_{time} contains request and response timing; and x_{vol} contains request and response sizes.

This tuple provides a coarse partition of the observable information at the client–RPC boundary by our threat model. The RPC handles sequences of requests from many senders over time. We model the full history as an interaction log.

Interaction Log. An *interaction log* is a time-ordered sequence of interaction tuples over functionalities and users: $R = (r_{F_1}, r_{F_2}, \ldots, r_{F_m})$.

Different RPC architectures reveal different portions of these interactions to infrastructure providers. We consider these providers as potential adversaries, and model what they observe through a *leakage function* L_{rpc} that maps from interaction information to the adversary's view.

Leakage Function, Record and Profile. For a fixed access architecture, such as MetaMask $\rightarrow$ Infura, the RPC *leakage function* L_{rpc} maps each interaction tuple r_F to the *leakage record* visible to the RPC as $L_{\mathsf{rpc}}(r_F) = (\ell_{\mathsf{id}}, \ell_{\mathsf{net}}, \ell_{\mathsf{app}}, \ell_{\mathsf{time}}, \ell_{\mathsf{vol}})$. Each component ℓ_c is the leakage from the corresponding interaction field x_c, based on the previously identified categories of leakage.

The adversary's power stems not just from observing individual leakages, but correlating their patterns over time. We refer to the sequence of leakage records that L_{rpc} produces over interaction log R as the architecture's *leakage profile*:

$L_{\mathsf{rpc}}(R) = (L_{\mathsf{rpc}}(r_{F_1}), L_{\mathsf{rpc}}(r_{F_2}), \ldots, L_{\mathsf{rpc}}(r_{F_m}))$.

A wallet workflow (Section 2) is analyzed by restricting R to the subsequence of interactions generated by that user-facing action and inspecting the leakage records of those interactions. The RPC receives no indication of which inter-

actions in R, generated by potentially many concurrent users, belong to the same user-facing action or workflow; grouping them into that subsequence is itself part of what an attack must accomplish (Section 4.2 instantiates this via burst-clustering).

Note that although ℓ_{net} may include transport/session metadata and browser/device fingerprints which are largely suppressible by standard client hygiene (clearing persistent storage, hardened uniform client stacks, TLS-fingerprint normalization), the rest of the paper takes $\ell_{\mathsf{net}} = \{\text{client IP}\}$.

For compactness, we decompose the application-layer leakage ℓ_{app} into the following context-specific components.

1. **addr$_{\mathbf{w}}$**: Sender's long-term wallet address exposed by writes such as a transaction envelope or payload.

2. **tx$_{\mathbf{w}}$**: Transaction parameters other than $\mathsf{addr}_{\mathsf{w}}$, including target contract, function selector, calldata fields, asset, value, route, amount, slippage bound, gas parameters.

3. **addr$_{\mathbf{r}}$**: Includes addresses, storage keys, transaction hashes and log filters exposed by queries of block chain data. The name $\mathsf{addr}_{\mathsf{r}}$ is shorthand because sender's long-term wallet address is the dominant case. These queries are reads since they don't change blockchain state.

4. **tx$_{\mathbf{r}}$**: Transaction parameters such as sender and destination addresses, value, gas fields exposed by a transaction simulation such as `eth_estimateGas`.

ℓ_{app} may additionally reveal coarse request metadata, most notably the invoked RPC method F (e.g. `eth_call` or `eth_getBalance`). We record F as residual leakage but do not assign it a separate component because none of the attacks studied here relies on F alone. We defer a finer-grained decomposition of F and a systematic analysis of other attacks that combine it with timing or volume to future work.

We show how the same leakage affects customers differently in Appendix B. Next, we show how existing architectures suppress leakage in Section 3.1.

3.1 Existing RPC Access Architectures

We apply our leakage framework to characterize concrete RPC access architectures in Table 1. Each row of this table states which components of $L_{\mathsf{rpc}}(R)$ are suppressed under an architecture's trust assumptions.

Non-private baseline. The dominant access architecture is account-bound API-key access: the RPC provider sees the client's identity details (name, email, billing information). Since the client interacts with the RPC in plaintext, it observes every component of the interaction: ℓ_{id}, ℓ_{net}, all ℓ_{app} components, ℓ_{time} and ℓ_{vol}. This is the developer default of every major provider (Infura, Alchemy, QuickNode, Ankr), handling billions of requests [110]. The API key stays linkable to the client even when downstream mitigations are adopted.

We call an architecture a mitigation primitive when it hides at least one component and describe it by the component it targets. No mitigation primitive is a complete defense in isolation: an encrypted mempool hides tx_{w} not $\mathsf{addr}_{\mathsf{r}}$ or tx_{r}, and PIR/ORAM hides $\mathsf{addr}_{\mathsf{r}}$ not ℓ_{net}, $\mathsf{addr}_{\mathsf{w}}$, tx_{w}, or tx_{r}.

Suppressing Identity (ℓ_{id}). Non-custodial wallets such as MetaMask, Phantom, and Rabby can be used without an RPC-provider account and form a widely used access path. MetaMask and Phantom report 30M+ and 15M monthly active users, respectively, and Rabby reports 4.2M installs [52, 55, 164]. However, wallet-mediated access need not introduce an independent, non-colluding party. For instance, MetaMask uses Infura by default, and both are Consensys products. [55, 149]. It has even been documented that Metamask sends the client's IP address to Infura [79].

Interactive consumer-wallet access is not a practical substitute for automated backend interfaces used by trading firms and indexers. MetaMask transaction submission ordinarily creates a user-confirmation request [147]; indexers connect directly to EVM JSON-RPC providers and may require archive state and `trace_filter` support [194]. MEV searchers use dedicated methods for bundle simulation and submission, such as `eth_callBundle` and `eth_sendBundle` [81, 86].

Two "privacy-enhancing architectures" actually suppress only ℓ_{id} when used without account-bound authentication. First, MEV-protect endpoints such as Flashbots Protect and MEV Blocker (2.1M Ethereum accounts and 4.5M+ users respectively), keep signed transactions out of the public mempool and route them through private order-flow systems to builders, but the receiving RPC sees the transactions in plaintext [58, 121]. Second, the Shutter encrypted-mempool deployment is on Gnosis Chain, not Ethereum L1; its documented encrypting-RPC both accepts plaintext transactions and encrypts them, so it remains trusted with their contents [184].

Suppressing Network identity (ℓ_{net}). The secure proxies of Section 2 suppress ℓ_{net}, but not every gateway-style intermediary qualifies: permissionless gateways such as POKT and Lava forward unencrypted JSON-RPC requests, so they observe ℓ_{app} and fail confidentiality [126, 166]. We surveyed the top 200 RPC services listed by CompareNodes [53] and found that the only deployed private RPC that eliminates ℓ_{net} by default is 1RPC, a TEE proxy, which self-reports 10M+ monthly users [16]. Unlinkability holds against the RPC only under proxy–RPC non-collusion, a strong assumption that often fails: an Infura-operated gateway forwarding to Infura's own RPC colludes by construction [109]. Chaining secure proxies (as in Tor and mixnets) mitigates documented single-proxy compromise [112, 143, 182, 195]. Beyond ℓ_{net}, mixnets and DC-nets also suppress ℓ_{time} and ℓ_{vol}, and the TEE oblivious mailbox suppresses ℓ_{time}. Time-perturbing mitigations trade privacy for latency: Loopix-style mixnets achieve second-scale latency [165], while user studies suggest that mean delays of 7–10 seconds degrade real-time usability [65]. Such mechanisms are poorly suited to latency-sensitive trading traf-

Table 1: RPC access architectures: deployment status, leakage profile and trust assumption

	Deployment status			Leakage Profile								
Technique	**Default**	**Opt-in**	**Research**	ℓ_{id}	ℓ_{net}	$\mathsf{addr_w}$	$\mathsf{tx_w}$	$\mathsf{addr_r}$	$\mathsf{tx_r}$	ℓ_{time}	ℓ_{vol}	**Trust**
API-key RPC [4, 11]	✓	×	×	×	×	×	×	×	×	×	×	RPC
Wallet default [52, 55, 164]	✓	×	×	✓	×	×	×	×	×	×	×	wallet/RPC
MEV-protect RPC [24, 58, 121]	×	✓	×	✓	×	×	×	×	×	×	×	receiving RPC
ERPC encrypted mempool [184]	×	✓	×	✓	×	×	×	×	×	×	×	ERPC + keypers
Permissionless gateway [126, 166]	×	✓	×	✓	✓	×	×	×	×	×	×	gateway
VPN [71, 182]	×	✓	×	✓	✓	×	×	×	×	×	×	VPN
Decentralized VPN [152, 160, 183]	×	✓	×	✓	✓	×	×	×	×	×	×	honest route
Tor [70, 199]	×	✓	×	✓	✓	×	×	×	×	×	×	guard/exit non-collusion
TEE relay [16]	×	✓	×	✓	✓	×	×	×	×	×	×	TEE
TEE oblivious mailbox [91]	×	×	✓	✓	✓	×	×	×	×	✓	×	TEE mailbox
DC-net [85, 177]	×	×	✓	✓	✓	×	×	×	×	✓	✓	anytrust/TEE
Oblivious HTTP [197]	×	✓	×	✓	✓	×	×	×	×	×	×	relay/gateway non-collusion
Mixnet [69, 106, 156]	×	✓	×	✓	✓	×	×	×	×	✓	✓	honest mix
Shielded pools [36, 62, 105, 205]	×	✓	×	✓	×	△	△	×	×	×	×	zk + anon. set
Stealth addresses [88, 210–212]	×	✓	×	✓	×	△	×	×	×	×	×	ECDH
Wallet-side encrypted mempool [78]	×	×	✓	✓	×	×	✓	×	×	×	×	t-of-n committee
Delay / timelock / VDF [3, 122]	×	×	✓	✓	×	×	✓	×	×	×	×	VDF / key release
PIR / ORAM reads [29, 103, 146, 226]	×	×	✓	✓	×	×	×	✓	×	×	×	crypto / non-collusion
TEE + ORAM reads [158]	×	×	✓	✓	×	×	×	✓	×	×	×	TEE + ORAM
TEE confidential gateway [82, 157, 181, 191]	×	×	✓	✓	×	×	✓	×	✓	×	×	TEE
Full-stack wallet SDK [124]	×	×	✓	✓	✓	✓	✓	✓	✓	△	△	composite

Under Leakage Profile, a ✓ means leakage component is suppressed; a × means it is visible; a △ means protection is partial. In the Deployment-status columns, a ✓ marks the applicable category and a × means it does not apply. ℓ_{id}, ℓ_{net}, $\mathsf{addr_w}$, $\mathsf{tx_w}$, $\mathsf{addr_r}$, and $\mathsf{tx_r}$ are as in Section 3. All rows except the first assume wallet-mediated access rather than account-bound API-key access.

fic; moreover, hiding the source IP offers little anonymity to customers already identified via RPC billing information.

Suppressing sender address in writes ($\mathsf{addr_w}$). Substituting a fresh per-transaction sender address can unlink a write from the user's long-term identity. A stealth-address recipient can later spend from its one-time address without exposing its long-term address, but funding, reuse, existing positions, or approvals can restore the link. For example, moving 100,000 USDC from long-term address A to fresh address S immediately before S trades exposes $A \to S$ to the RPC. Stealth addresses [210, 211] see negligible use: Umbra covers only 0.002% of Ethereum transactions [212], and Fluidkey supports only 24K users [88]. Shielded pools such as Railgun and Hinkal (∼4B [62] and ∼400M [105] in volume) hide the sender, but correlated amounts can link the two ends; for example, depositing 10 ETH from A and later withdrawing 9.97 ETH to B suggests an $A \leftrightarrow B$ match [190]. Thus, both mechanisms suppress $\mathsf{addr_w}$ only approximately.

Suppressing transaction parameters in writes ($\mathsf{tx_w}$). Hiding transaction submission parameters, the exact input an RPC needs to carry out a sandwich MEV attack [46], requires the wallet to hand the RPC a ciphertext that decrypts on-chain only after ordering is fixed, as standardized in the Universal Enshrined Encrypted Mempool proposal [75]. Both *wallet-side encrypted mempools* [78] and *delay/timelock/VDF* schemes [122] (Section 2) are research-stage and require wallet-side encryption. Aptos's encrypted mempool is designed specifically for the Aptos chain; Ethereum L1 does not currently deploy Aptos's mechanism. Because plaintext simulations still expose $\mathsf{tx_r}$, encrypted mempools alone do not prevent sandwich attacks.

Suppressing query index in reads ($\mathsf{addr_r}$). *PIR/ORAM* and

TEE+ORAM [103, 146, 158] hide the queried index ($\mathsf{addr_r}$) in reads from the RPC. Both are research-stage, since current PIR/ORAM constructions do not scale to Ethereum's ∼245 GiB live state [185] and have been demonstrated only for permissioned ledgers or smaller shards [15]. State-of-the-art oblivious storage systems such as Snoopy require a cluster of 18 SGX-enabled machines to sustain 92K requests/s over just two million 160-byte objects [63], several orders of magnitude short of Ethereum's live state and request volume, so scaling such a design to production RPC traffic would multiply that hardware footprint accordingly. Because these reads leave no on-chain trace [102], suppressing $\mathsf{addr_r}$ denies the RPC a vantage unavailable to on-chain observers, important especially when ℓ_{id} already identifies the customer.

Suppressing transaction simulation contents ($\mathsf{tx_r}$). Hiding simulation contents has two paths: (a) evaluating `eth_call` on encrypted parameters via server-side secure computation (MPC, FHE, or TEE-based execution), or (b) running the simulation client-side on a light client, so the RPC never receives the call at all. *TEE confidential gateways* realize path (a) on non-Ethereum chains such as Sapphire, Secret Network and TEN, and Ethereum's builder layer (BuilderNet), but adapting them to the Ethereum RPC layer remains open. MPC/FHE variants remain research-stage [82, 157, 181, 191].

3.2 Leakage-Class Taxonomy (Type 0 to 6)

Rather than enumerate every subset of suppressible leakage components, we compose the primitives into seven leakage classes whose RPC-visible leakage decreases cumulatively: Type i suppresses every component suppressed at Type $i-1$, plus one additional component. This ordering is expository, it follows the additional infrastructure and engineering required on the ordinary Ethereum wallet path; in PIR for instance, this is directly quantified as measurable cloud-compute and bandwidth cost per query [5]. Real deployments need not fit this order; API-key RPC over a VPN hides ℓ_{net} without removing ℓ_{id}. Architectures requiring only client-side logic or an extra hop are deployed defaults (Types 0–1) or opt-in (Types 2–3) today, while primitives requiring dedicated infrastructure or custom cryptographic protocols, such as PIR/ORAM, remain research-stage (Types 4–6). The hierarchy orders ℓ_{id}, ℓ_{net}, and the four decomposed ℓ_{app} components leaving F, ℓ_{time} and ℓ_{vol} visible at Type 6, which can be suppressed at Type $6^\star$ via techniques like ORAM and DC-nets.

Table 2 gives $L_{\mathsf{rpc}}(R)$ at each class. For Types 1 through 6, the corresponding mechanisms as an example are respectively, wallets, a secure proxy, stealth addresses or shielded pools (only approximately), wallet-side encrypted mempools, PIR/ORAM or TEE+ORAM reads, and TEE confidential gateways. At Type 2 the proxy and RPC see complementary halves of the interaction (Fig. 2), so the table also lists the proxy's view $L_{\mathsf{P}}(R)$. For compactness, components suppressed at earlier Types are omitted from later rows; $\perp$ marks

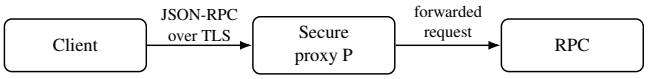


Figure 2: In Type 2, secure proxy observes client's network identity, not JSON-RPC contents; RPC observes forwarded request contents, not client's source IP.

Table 2: Cumulative Type hierarchy: leakage profile $L_{\mathsf{rpc}}(R)$, plus secure proxy view $L_{\mathsf{P}}(R)$ at Type 2.

Type	$L_{\mathsf{rpc}}(R)$
0	$(\ell_{\mathsf{id}}, \ell_{\mathsf{net}}, \ell_{\mathsf{app}}, \ell_{\mathsf{time}}, \ell_{\mathsf{vol}})$
1	$(\ell_{\mathsf{id}}{=}\perp, \ell_{\mathsf{net}}, \ell_{\mathsf{app}}, \ell_{\mathsf{time}}, \ell_{\mathsf{vol}})$
2	$(\ell_{\mathsf{net}}{=}\perp, \ell_{\mathsf{app}}, \ell_{\mathsf{time}}, \ell_{\mathsf{vol}})$
	$L_{\mathsf{P}}(R){=}(\ell_{\mathsf{net}}{=}\{\mathsf{IP}\}, \ell_{\mathsf{app}}{=}\perp, \ell_{\mathsf{time}}, \ell_{\mathsf{vol}})$
3	$(\ell_{\mathsf{app}}{=}\{\mathsf{addr_w}{=}\perp, \mathsf{tx_w}, \mathsf{addr_r}, \mathsf{tx_r}\}, \ell_{\mathsf{time}}, \ell_{\mathsf{vol}})$
4	$(\ell_{\mathsf{app}}{=}\{\mathsf{tx_w}{=}\perp, \mathsf{addr_r}, \mathsf{tx_r}\}, \ell_{\mathsf{time}}, \ell_{\mathsf{vol}})$
5	$(\ell_{\mathsf{app}}{=}\{\mathsf{addr_r}{=}\perp, \mathsf{tx_r}\}, \ell_{\mathsf{time}}, \ell_{\mathsf{vol}})$
6	$(\ell_{\mathsf{app}}{=}\{\mathsf{tx_r}=\perp, F\}, \ell_{\mathsf{time}}, \ell_{\mathsf{vol}})$
$6^\star$	$(\ell_{\mathsf{app}}=\perp, \ell_{\mathsf{time}}=\perp, \ell_{\mathsf{vol}}=\perp)$

the component newly suppressed at the current Type.

Under our taxonomy, Kohaku [76, 124] is designed to compose mechanisms spanning the Type 0 to 6 hierarchy in one wallet stack. Type 0 is the unprotected baseline, and Type 1 requires only accountless wallet-mediated access; the remaining Types add potential pluggable support for Flashnet [33, 85] for Type 2-style network anonymity in transaction submission, stealth addresses and Railgun shielded pool for Type 3, planned wallet-side threshold encryption for Type 4, TEE+ORAM/PIR for Type 5, and a browser-side minimal execution client for transaction simulations in Type 6.

Concretely, our hierarchy aligns with Apple's four-level maturity model for cloud-AI privacy [125]. Type 0 matches Apple's Level 0; thereafter, the hierarchies suppress identity and content in opposite orders: Types 1 and 2 suppress identity first and Type 6 suppresses the four decomposed application-layer components, whereas Apple's Level 1 protects content and Level 2 provides non-targetability i.e. identity suppression. Type 6 is therefore closest to Apple's Level 2 in leakage terms, while Types 4, 5 and $6^\star$ have no PCC analogues because PCC does not model transaction parameters, read-query indices, or timing/volume leakage. PCC further provides verifiable transparency and defenses against physical and supply-chain attacks; these integrity properties fall outside our leakage hierarchy and are discussed in Section 5.

4 Leakage-Abuse Attacks

Leakage profiles describe what each RPC architecture exposes but not the real-world impact of that exposure. Section 4.1 develops an attack framework that turns a leakage profile into

a concrete attack, which we then instantiate for deanonymization (Section 4.2), censorship (Section 4.3), and sandwich MEV (Section 4.4). Together these instantiations measure the extent and financial impact of privacy compromise.

4.1 Our Attack Framework

Passive and active adversaries. Every RPC adversary is at least *passive*: it follows the protocol and only observes the leakage profile $L_{\mathsf{rpc}}(R)$. This alone enables deanonymization. The other attacks require the RPC to *actively* deviate from the protocol, which we capture as four *active capabilities*: C_{drop} suppresses a chosen request or transaction, C_{delay} delays it arbitrarily, C_{inject} inserts an adversarially chosen one, and C_{route} forwards it to an adversarially preferred downstream party or route. We model an RPC adversary by the blockchain intermediaries it colludes with and the active deviations it can perform, beyond its ordinary leakage observation.

Adversary Model. An adversary model M defines the set of parties the RPC colludes with, and the set of active capabilities. If the RPC has no active capabilities, we consider it a passive adversary; otherwise it is an active adversary. Unless stated otherwise, we analyze an RPC adversary that does not collude with any other party. Having fixed the adversary model, we describe each attack by specifying what side information the adversary has, how it acts on the leakage profile, and what output or effect it seeks to produce.

Attack. An attack is a tuple $\Pi = (M, K, A, G)$. Here, K is prior knowledge available to the RPC-adversary, such as known contract addresses, public social-media data, sanctions lists, geolocation databases, protocol parameters, or public-chain history. A is a probabilistic polynomial-time algorithm that takes $(L_{\mathsf{rpc}}(R), M, K)$ as input and produces the attack output G. For each attack we instantiate the smallest set of active capabilities in M, sufficient to realize the goal G; granting the adversary additional capabilities only strengthens the attack.

Evaluation preview. We instantiate $\Pi = (M, K, A, G)$ for each attack below, with full pseudocode in Appendix E. A varies by Type only for deanonymization; for censorship and MEV, A is fixed, while the Type determines which predicates remain computable. Appendix D implements each attack on our interaction logs and tests whether the leakage at each Type suffices to carry it out.

4.2 Deanonymization

The deanonymization attack is $\Pi_{\mathsf{deanon}} = (M, K, A, G)$ where the adversary model M represents a passive RPC provider. The goal G is to partition observed addresses into clusters and assign each cluster to a user, recovering their portfolio of owned addresses and balances.

Metrics. We evaluate this attack along three dimensions. $\mathsf{Accuracy}$ is the fraction of addresses in S assigned to the correct user. $\mathsf{Precision}(u)$ is the fraction of addresses attributed to user u that truly belong to u. $\mathsf{Recall}(u)$ is the fraction of user u's true addresses that the attack recovers.

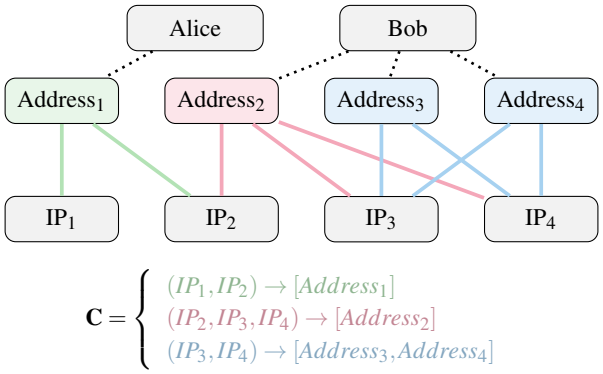


Figure 3: Observed address–IP graph: Alice controls Address_1; Bob controls Addresses 2, 3, and 4. The clustering C identifies three clusters based on the IP fingerprinting.

We now initialize the attack algorithm A for each architecture type, along with prior knowledge K.

Type 0. Here, no auxiliary knowledge K is assumed. First, the client–RPC interactions are clustered by API key or user account identifier. Each cluster corresponds to a user, and the set of addresses in each cluster represents the addresses owned by the user. The public ledger is queried to obtain the balance of every address, therefore obtaining the balance of every user. The attack is direct, by construction, this gives perfect accuracy, per-user precision and recall.

Type 1. The prior knowledge K is the set of IP addresses associated with each user, realistic as a coarse version of this knowledge is obtainable from IP geolocation data, repeated visits, or auxiliary logs from dApps, exchanges, analytics services, support systems, or prior authenticated sessions [113, 142]. This strengthens deanonymization in the face of IP churn and shared egress such as NAT and public Wi-Fi. First, the attack constructs the set of IP addresses i.e. the IP signature, corresponding to each address is obtained from the client–RPC interactions. Addresses with nearly the same IP signature are clustered together as in Fig. 3. These clusters are refined by dropping transient IPs, splitting on device/TLS/session fingerprints when available, and merging clusters with identical address sets. Each of these clusters corresponds to a user that is associated with the superset of IPs in the cluster's IP set.

Type 2-4. In a portfolio-refresh workflow, reads arrive as a tight burst Fig. 4, allowing A to group addresses whose $\mathsf{addr_r}$ queries arrive within the same interval using only $\mathsf{addr_r}$ and ℓ_{time}. The prior knowledge K is a list of candidate users, along with their estimated total on-chain wealth, but not yet linked to any specific address. Such lists are realistic: crypto rich lists and analytics platforms already estimate holdings of high-profile individuals from public disclosures, journalistic

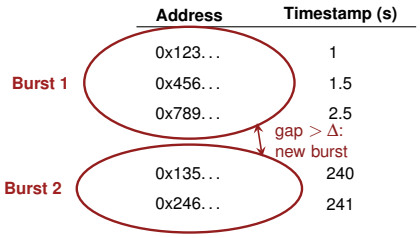


Figure 4: Balance reads clustered into temporal bursts with gap threshold $\Delta = 1$ s.

investigation, blockchain data, court records, social-media-linked wallet tags, and exchange records obtained through legal process [89, 174, 175, 203, 204]. This knowledge allows the RPC to map each cluster of addresses to the right user.

Type 5-6. $\mathsf{addr_r}$ is hidden, so the RPC no longer sees which addresses a user queries. Deanonymization falls back to the public-ledger analysis available to any on-chain observer.

Deanonymization enables targeted poisoning. In the 2026 KelpDAO bridge incident, compromised LayerZero RPC nodes answered LayerZero's monitors correctly but falsely reported only to the LayerZero labs' Decentralized Verifier Network (DVN) signing service that rsETH had been burned on the source chain, avoiding the inconsistency that uniform poisoning would expose [40, 127]. The incident shows how an RPC provider with response-forging capability could similarly poison a deanonymized wallet while returning correct responses to other clients, including monitors.

4.3 Censorship

The censorship attack is $\Pi_{\mathsf{censor}} = (M, K, A, G)$, where the adversary model M represents an RPC provider capable of dropping any request (C_{drop}). The auxiliary knowledge K includes the list of sanctioned users, set of blocked jurisdictions based on IP range, and a set of blocked smart contract addresses (e.g. Tornado Cash [205]), and the address-to-user map obtained by deanonymization.

The adversary evaluates three predicates, each indexed by the leakage component consumed. The *entity* predicate fires when the sender address, after deanonymization, resolves to a user in the sanctioned set. The *jurisdiction* predicate fires when the source IP falls within a blocked jurisdiction. The *contract* predicate fires when the destination address corresponds to a blocked smart contract. An interaction is targeted if *any* predicate fires. The goal G is to drop every targeted interaction and forward the rest. These predicates correspond to censorship vectors documented in prior incidents [209].

Metrics. We measure censorship along three dimensions. Selectivity is the fraction of all interactions that the adversary drops. Accuracy is the fraction of targeted interactions that were actually dropped. The false-positive rate FPR is the fraction of non-target interactions dropped.

We go Type by Type through the entity, jurisdiction, and contract predicates: as leakage is hidden, each works on weaker information and loses accuracy.

Type 0. All three predicates work perfectly: the RPC drops exactly the targeted interactions, nothing else.

Type 1. The jurisdiction predicate (from ℓ_{net}) and contract predicate (from $\mathsf{tx_w}$) are exact, and the entity predicate follows from Type 1 deanonymization.

Type 2. The jurisdiction predicate degrades: the source IP is now the secure proxy's exit IP (Section 3.1), resolving to the proxy's jurisdiction, not the user's. The entity predicate still works via Type 2 deanonymization, and the contract predicate persists whenever the destination contract appears in $\mathsf{tx_w}$ or a preceding $\mathsf{tx_r}$ simulation.

Type 3. Although $\mathsf{addr_w}$ is suppressed, linking the write to its preceding portfolio-refresh burst exposes $\mathsf{addr_r}$, enabling deanonymization; the entity predicate still works. Contract predicate is intact, since destination contract stays in $\mathsf{tx_w}$.

Type 4. Although $\mathsf{tx_w}$ hides the submitted transaction's destination, preceding `eth_call` and `eth_estimateGas` simulations expose it through $\mathsf{tx_r}$, preserving the contract predicate. Entity predicate remains successful via $\mathsf{addr_r}$.

Type 5. With $\mathsf{addr_r}$ hidden, the entity predicate fails. Contract predicate is as in Type 4. So accuracy falls relative to Type 4.

Type 6. Hiding $\mathsf{tx_r}$ closes the contract predicate too. With no predicate left to fire on, censorship loses all targeting and reduces to availability degradation.

Censorship is selective. Mistargeted drops produce user-visible failures and wallet-side error reports, raising detection and reputation costs; selective censorship is therefore plausible only at tolerably low false-positive rates.

4.4 Sandwich MEV

Deanonymization and censorship require *selective* targeting; sandwich MEV instead requires two conditions, which are our two evaluation dimensions: *visibility* into the victim swap's parameters ($\mathsf{tx_w}$, captured by the leakage classes), and sufficient *ordering control* to place attacker transactions around it, modeled as *sequencing noise*.

Sequencing noise. Transaction parameter visibility is necessary but not sufficient for realized sandwich profit. Assuming the front-run (tx_F), victim (tx_V), and back-run (tx_B) transactions are all included in the block, the realized profit further depends on what other transactions land alongside them and in what order. Throughout, suppose the victim and the front-run tx_F swap token X for Y, while the back-run tx_B swaps Y back for X. We define *sequencing noise* as the deviation of the realized intra-block order from the attacker's intended sequence (tx_F, tx_V, tx_B) (Fig. 5), and distinguish two types:

Interleaving noise: the relative order $tx_F \prec tx_V \prec tx_B$ is preserved, but unrelated transactions land between tx_F, tx_V, and tx_B, updating pool reserves before tx_B executes.

Reordering noise: the relative order $tx_F \prec tx_V \prec tx_B$ is not preserved; the three may land in any of the six permutations.

We evaluate the attack under three execution settings, each corresponding to a distinct combination of noise types:

E_0 (no noise): neither interleaving nor reordering noise is present. This regime reflects the PBS path assumed by most recent sandwich-MEV evaluations [46, 101, 171]: the attacker submits (tx_F, tx_V, tx_B) as a single bundle via `eth_sendBundle` to a Flashbots-compatible builder, which preserves bundle atomicity [81] (the triple if included, is in order with no transactions in between), and the block wins the MEV-Boost auction [84] . It caps *per-victim* sandwich profit, but not the attacker's total profit, which interleaving noise in E_1 can push even higher.

E_1 (interleaving noise only): the attacker submits tx_F, tx_V, tx_B independently with descending priority fees $p_F > p_V > p_B$, and the builder orders by descending effective gas price (Flashbots `greedy` [80]). Any unrelated transaction with a priority fee between p_F and p_V or p_V and p_B then interleaves between tx_F and tx_B, preserving the attacker's relative order.

Interleaving helps or hurts the attacker depending on the inserted swaps' direction. A same-direction swap (another $X \to Y$, like tx_F) pushes the price further along the curve, adding to the impact tx_B later captures, so the attacker earns more than with no noise (*substitution*). An opposite-direction swap ($Y \to X$) instead pushes the price back, partly undoing tx_F's impact and leaving tx_B less to recapture (*dilution*). Empirically, a sandwich's front and back-run transactions are often separated by several intervening ones [171, 200], consistent with substitution.

E_2 (interleaving and reordering noise): both noise types are present, so attacker's intended $tx_F \prec tx_V \prec tx_B$ order is no longer preserved. This is the goal of anti-sandwich defenses: order-fairness consensus [119] and random-permutation ordering such as Alpos et al.'s Π^3 [6] scramble the ordering precisely so the attacker cannot reliably place its front and back-run around the victim.

The sandwich MEV attack is $\Pi_{\mathsf{mev}} = (M, K, A, G)$, where the adversary model M represents an RPC capable of dropping any transaction (C_{drop}), delaying its transmission (C_{delay}), injecting its own transaction on-chain (C_{inject}) and forwarding transactions to preferred builders (C_{route}). The auxiliary knowledge K consists of the constant-product pool reserves (r_X, r_Y) and fee rate φ (Uniswap-v2 uses $\varphi = 0.3\%$), both publicly readable. Given victim's input amount Δ_{in} and minimum acceptable output $\Delta_{\mathsf{out}}^{\mathsf{min}}$ (derived from slippage tolerance), the goal G is to realize sandwich profit.

The adversary evaluates a single *profit* predicate, indexed by the leakage component it consumes. It fires on a swap when the maximum realizable sandwich profit is strictly positive, computable from $\mathsf{tx_w}$ or, when hidden, from the $\mathsf{tx_r}$ in a preceding `eth_call` or `eth_estimateGas` simulation. For each flagged swap, the attacker binary-searches for the largest front-run input keeping the victim's output above $\Delta_{\mathsf{out}}^{\mathsf{min}}$,

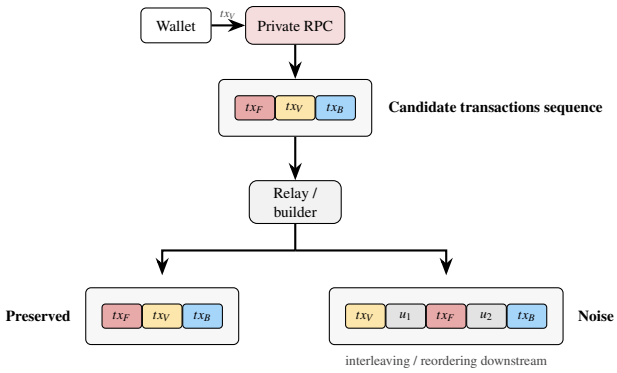


Figure 5: Private RPC constructs a candidate sandwich transaction sequence $\langle tx_F, tx_V, tx_B \rangle$ and forwards to the relay/builder. Downstream ordering may either preserve or disrupt the transaction sequence through reordering and interleaving with unrelated transactions u_i.

computes the back-run output and net profit (after gas, the priority-fee premium, and any builder or relay payment to secure ordering), and executes the sandwich if that profit is positive. Sandwich attacks are pairwise-local in standard AMM models, so one sandwich per victim gives the per-block upper bound: the RPC's opportunity is the sum over profitable victim swaps it observes in the slot [46].

Metrics. For a target set T of victim swaps, aggregate attacker profit $\mathsf{Profit}(T)$ sums realized attacker profit across T, while $\mathsf{ProfRate}(T)$ is the fraction of T yielding strictly positive profit. Both include victim-revert cases in which same-direction interleaving supplies the price impact captured by tx_B. Note that the attacker profit denotes the upper bound on RPC-realized profit, not the actual profit. We now go Type by Type, combining leakage class with execution setting.

Type 0–3. $\mathsf{tx_w}$ makes the profit predicate exactly computable: E_0 attains the per-victim upper bound, E_1 changes profit through substitution or dilution, and E_2 removes reliable ordering and can make aggregate profit negative.

Type 4–5. Although $\mathsf{tx_w}$ is hidden, the predicate remains computable from $\mathsf{tx_r}$ exposed by preceding `eth_call` or `eth_estimateGas` simulations.

Type 6. With $\mathsf{tx_r}$ hidden too, the profit predicate is uncomputable from $L_{\mathsf{rpc}}(R)$, closing the attack surface.

A realized sandwich requires both swap parameters visibility and transaction ordering control; removing either prevents it; we refer to this mechanism as an MEV countermeasure.

MEV Countermeasure. A *countermeasure* must reach (a) Type 6 or (b) E_2. Execution-rule changes that make per-victim profit non-positive (e.g. batch auctions, uniform clearing [224]) are out of scope, since they do not apply to existing AMMs like Uniswap. Table 5 in Appendix C classifies deployed and proposed countermeasures [225, 228] along the two axes. Notably, the most widely adopted class—private-routing platforms like MEV-Boost [84] and Flashbots Pro-

tect [83]—still exposes tx_w to the receiving RPC, so it remains Type 0-3 with E_0 available. Fair ordering alone is provably insufficient against arbitrary payoff functions [46], ruling out the only E_2 mitigation.

5 Conclusion and Future Work

The leakage-and-attack framework we propose is designed to be extensible, and the instantiation in this paper is deliberately bounded: the leakage components we model are those our attacks consume, the network model reduces ℓ_{net} to the client IP, the attack instantiations cover three classes, and the evaluation replays constructed workflows rather than production traffic. The framework is also privacy-centric and analyzes each RPC in isolation. These boundaries motivate the directions below.

Customer inference validation on production traffic. Appendix B derives customer-specific inferences from RPC leakage, but provider confidentiality prevents validation on production traces. Future work should test these inferences against independently verified ground truth and report attack success by customer class. Bridges and autonomous agents merit dedicated study. Bridge RPC traffic may reveal destination-chain transaction details before reaching the destination mempool [129]. For autonomous agents, their multi-step reads, simulations, and submissions may reveal their delegated task or authorization conditions [8, 176].

Attacks in Type 6 architecture. Type 6 retains functionality F, request timing ℓ_{time}, and request volume ℓ_{vol}, but attacks using these components without identity or transaction contents remain open. A possible attack we infer from [7] is that the RPC need not read the sealed bid in a sealed-bid auction: the timing and mere existence of bid transactions in the auction window suffice to drop or delay competitors at the entry point, leaving only its own bid included to win. Wu et al. estimate CEX-DEX searchers' off-chain hedge timing from on-chain arbitrage transactions and CEX prices [221]. Whether Type 6 traces alone reveal trading workflows or proprietary strategy information requires empirical study.

Attacks based on Network identity beyond IP address. Hiding a public IP may not eliminate network-layer linkability: IPv4 IP-ID generation [123], stable IPv6 identifiers [56], TLS ClientHello fingerprints [92], and session-resumption tickets [189] can identify returning devices or client implementations across connections. Future work should measure whether these signals re-link wallets across IP changes and therefore remain part of ℓ_{net}.

Evaluating realistic poisoning attacks. The KelpDAO incident [40, 127] demonstrates that compromised RPC nodes can selectively forge responses. A systematic cross-provider study of this behaviour remains missing. Independently verifiable probes, analogous to Spoiled Onions' decoy traffic to detect malicious Tor relays [219] could detect indiscriminate poisoning. Detecting selective poisoning additionally requires targeted probes that reproduce realistic victim profiles based on which RPCs could poison requests.

Integrity mitigations. Our survey is privacy-centric; and systematizing the integrity guarantees of RPC architectures remains open. The following existing mitigations for instance don't guarantee true integrity.

Proof-carrying responses. `eth_getProof` [115] authenticates a value only relative to a trusted state root; obtaining both from one RPC permits a malicious provider to return a poisoned root with a consistent proof. Quorum-based verification offers an alternative [116, 192], but shared infrastructure or software can undermine provider independence, as illustrated in the October 2025 AWS us-east-1 outage [50]. Future work should formalize independence and collusion assumptions and characterize the guarantee obtained by combining quorums with proof-carrying responses for RPCs.

TEE-hosted RPC. Ankr's Verifiable RPC attests the software producing each signed response [10]. Side-channel attacks like Foreshadow [206] and SGAxe [207] that extract attestation-related keys motivate avoiding a single hardware trust root; Apple's Private Cloud Compute instead uses multiple hardware vendors [12]. Whether RPCs can combine node-level attestation with multi-vendor roots and a threat model covering side-channels remains open.

Beyond RPCs as adversaries. Our leakage framework can be extended to any intermediary in the blockchain ecosystem, not just RPCs: wallets and dApp front ends [79, 215], centralized L2 sequencers, and exchanges all occupy this position. Sequencers can also reorder or delay transactions and extract value from them. Common ownership may allow observations across services to be combined: Coinbase operates an RPC service and sequences Base [49], OKX sequences and serves the public RPC for X Layer [159], and MetaMask defaults to its parent company Consensys' RPC service, Infura [55, 149]. Appendix A extends our framework to such joint views. Applying this extension to deployed systems and analyzing the resulting attacks remain future work.

Anonymous payment/subscription with RPC providers. Request-level privacy does not prevent credit cards, bank accounts, or KYC-linked payments from associating RPC usage with a durable identity. Privacy Pass, Anonymous Credit Tokens, and ZK API Usage Credits provide unlinkable authorization or usage-based payment mechanisms, with the latter explicitly targeting Ethereum RPC access [59, 64, 179]. Their deployment should be evaluated jointly with network and application privacy because deposit timing and small anonymity sets may still link requests to customers.

Latency–privacy tradeoff. Type 5 and 6 architectures are still research-stage and don't yet meet customer-specific RPC performance requirements. Future architectures should be evaluated under realistic RPC workloads, latency requirements, and provider trust assumptions.

Ethical Considerations

All measurements in this paper are performed on data we generate ourselves or on public, already-confirmed on-chain records. We replay public, already-confirmed Ethereum transactions for 550 simulated users through a self-hosted Ganache [202] RPC endpoint. We use no production RPC logs, and we did not probe, measure, or attack any commercial RPC provider: the shared-IP and Tor-exit configurations are synthetic labels applied to our own trace rather than real network-layer activity. The sandwich-MEV evaluation likewise replays public, already-confirmed Uniswap-v2 swaps offline against simulated pool reserves; the front-run and back-run transactions we construct are never broadcast to Ethereum mainnet, so no value is extracted from any real party. The USD totals quantify simulated extractable value under our stated market model, not historical extraction opportunities.

The attacks we implement are therefore executed by us in the role of the RPC against our own interaction log; no real user is deanonymized, censored, or sandwiched. As a result, our results identify no vulnerability that warrants coordinated disclosure. Where we describe an incident in which forged RPC responses were exploited in practice, we rely solely on published incident reports [40, 127], and our characterization of named providers rests on their public documentation and privacy policies. The cross-provider probing we identify as future work in Section 5, which would send verifiable probes to live providers to detect selective response poisoning, would require its own ethical review; we do not undertake it here.

Open Science

We provide an anonymized artifact at https://anonymous.4open.science/r/Blockchain_Centralization_RPC-E355/README.md. It contains the pipeline that produces the client–RPC interaction dataset used in Appendix D, implementations of the deanonymization, censorship, and sandwich-MEV attacks, and the configurations and analysis scripts used to produce the reported results. This interaction log is derived from public Ethereum data. The accompanying README documents the required environment, commands to reproduce what we report, and correspondence between artifact outputs and the paper's tables and figures. All artifacts will be made publicly available under a stable, non-anonymous link upon acceptance. The systematization is based on the publicly available papers, specifications, and deployment documentation cited in the paper; no non-public evidence was used.

References

[1] Hayden Adams, Noah Zinsmeister, and Dan Robinson. Uniswap v2 Core. https://uniswap.org/whitepaper.pdf, 2020.

[2] Amit Agarwal, Kushal Babel, Sourav Das, Babak Poorebrahim Gilkalaye, Arup Mondal, Benny Pinkas, Peter Rindal, and Aayush Yadav. Weighted batched threshold encryption with applications to mempool privacy. Cryptology ePrint Archive, Paper 2025/2115, 2025. URL: https://eprint.iacr.org/2025/2115.

[3] S. Agrawal and Vinay J. Ribeiro. Timelock Shield: A Robust Defense Against MEV and Censorship Attacks in Blockchain. In *2025 Crypto Valley Conference (CVC)*. IEEE, 2025.

[4] Alchemy. Alchemy platform: Web3 developer platform and infrastructure metrics. https://www.alchemy.com/, 2026. Reports infrastructure across 100+ networks and 100M+ end users. Accessed: 2026-05-09.

[5] Asra Ali, Tancrède Lepoint, Sarvar Patel, Mariana Raykova, Phillipp Schoppmann, Karn Seth, and Kevin Yeo. Communication–computation trade-offs in PIR. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1811–1828. USENIX Association, 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/ali.

[6] Orestis Alpos, Ignacio Amores-Sesar, Christian Cachin, and Michelle Yeo. Eating sandwiches: Modular and lightweight elimination of transaction reordering attacks. In *27th International Conference on Principles of Distributed Systems (OPODIS 2023)*, volume 286 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.OPODIS.2023.12.

[7] Orestis Alpos, Lioba Heimbach, Kartik Nayak, and Sarisht Wadhwa. Censorship-resistant sealed-bid auctions on blockchains. *Cryptology ePrint Archive*, 2025.

[8] Saad Alqithami. Autonomous agents on blockchains: Standards, execution models, and trust boundaries. https://arxiv.org/abs/2601.04583, 2026. arXiv:2601.04583.

[9] Elli Androulaki, Ghassan O. Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating user privacy in bitcoin. In *Financial Cryptography and Data Security*, 2013.

[10] Ankr. Verifiable RPC (vRPC): Cryptographically attested JSON-RPC. https://www.ankr.com/web3-api/verifiable-rpc/, 2026. Accessed: 2026-08-25.

[11] Ankr Inc. Ankr Privacy Policy. https://www.ankr.com/privacy-policy/. Last modified March 03, 2023. Accessed: 2026-05-05.

[12] Apple Security Engineering and Architecture. Expanding Private Cloud Compute. https://security.apple.com/blog/expanding-pcc/, June 2026. Apple Security Research blog, published June 8, 2026.

[13] Arbitrum DAO. Constitutional AIP: Proposal to adopt timeboost, a new transaction ordering policy. https://forum.arbitrum.foundation/t/constitutional-aip-proposal-to-adopt-timeboost-a-new-transaction-ordering-policy/25167, 2024. Finalized AIP; accessed 2026-05-08.

[14] Astria. Shared sequencer documentation. https://docs.astria.org/, 2025. Accessed: 2026-05-09.

[15] Ali Atiia and Keewoo Lee. Sharded PIR design for the Ethereum state. https://ethresear.ch/t/sharded-pir-design-for-the-ethereum-state/24552, 2026. Ethereum Research forum post, published March 30, 2026; accessed 2026-05-08.

[16] Automata Network. 1RPC: Privacy-preserving rpc relay. https://docs.ata.network/backed-by-pom/1rpc, 2026. Accessed: 2026-05-09.

[17] Aztec Network. Aztec connect sunset announcement. https://aztec.network/blog/sunsetting-aztec-connect, 2023. Accessed: 2026-05-09.

[18] Kushal Babel, Nerla Jean-Louis, Yan Ji, Ujval Misra, Mahimna Kelkar, Kosala Yapa Mudiyanselage, Andrew Miller, and Ari Juels. PROF: Protected order flow in a profit-seeking world. Cryptology ePrint Archive, Report 2024/1241, 2024. URL: https://eprint.iacr.org/2024/1241.

[19] Maryam Bahrani, Pranav Garimidi, and Tim Roughgarden. Centralization in block-building and proposer-builder separation. In *Financial Cryptography and Data Security (FC)*, Lecture Notes in Computer Science, pages 331–349. Springer, 2024. doi:10.1007/978-3-031-78676-1_19.

[20] Joseph Bebel and Dev Ojha. Ferveo: Threshold Decryption for Mempool Privacy in BFT Networks. Cryptology ePrint Archive, Paper 2022/898, 2022. URL: https://eprint.iacr.org/2022/898.

[21] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474, 2014. doi:10.1109/SP.2014.36.

[22] Ferenc Béres, István A Seres, András A Benczúr, and Mikerah Quintyne-Collins. Blockchain is watching you: Profiling and deanonymizing ethereum users. In *2021 IEEE international conference on decentralized applications and infrastructures (DAPPS)*, pages 69–78. IEEE, 2021.

[23] Blockworks. Consensys invites big tech to decentralized infrastructure network. https://blockworks.co/news/infura-decentralized-infrastructure-network, 2023.

[24] bloXroute Labs. Protect rpc documentation. https://docs.bloxroute.com/, 2026. Accessed: 2026-05-09.

[25] Alexandra Boldyreva, Zichen Gui, and Bogdan Warinschi. Understanding leakage in searchable encryption: A quantitative approach. *Proceedings on Privacy Enhancing Technologies*, 2024(4):503–524, 2024.

[26] Dan Boneh. CS 251: Blockchain Technologies. https://cs251.stanford.edu/syllabus.html, 2025. Stanford University course syllabus and lecture materials. Accessed: 2026-05-05.

[27] Dan Boneh, Rohit Nema, Arnab Roy, and Ertem Nusret Tas. Efficient batch threshold encryption using partial fraction techniques. *Cryptology ePrint Archive*, (2026/674), 2026. URL: https://eprint.iacr.org/2026/674.

[28] Cynthia Brumfield. Behind the coinbase breach: Bribery emerges as enterprise threat, August 2025. URL: https://www.csoonline.com/article/4042522/behind-the-coinbase-breach-bribery-is-an-emerging-enterprise-threat.html.

[29] Alexander Burton, Samir Jordan Menon, and David J. Wu. Respire: High-rate PIR for databases with small records. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1463–1477. Association for Computing Machinery, 2024. URL: https://eprint.iacr.org/2024/1165, doi:10.1145/3658644.3690328.

[30] Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform. White paper, 2014.

[31] Vitalik Buterin. Proposer-builder separation friendly fee market designs. Ethereum Research Forum. https://ethresear.ch/t/proposer-block-builder-separation-friendly-fee-market-designs/9725, 2021. Accessed: 2026-05-09.

[32] Vitalik Buterin. A local-node-favoring delta to the scaling roadmap. Ethereum Research Forum. https://ethresear.ch/t/a-local-node-favoring-delta-to-the-scaling-roadmap/22368, 2025. Accessed: 2026-05-09.

[33] Vitalik Buterin. On network-layer anonymization for the block-building pipeline. https://x.com/VitalikButerin/status/2028524112868708616, 2026. Discusses Flashnet as a latency-minimized network-layer anonymization design and states the Kohaku initiative is expected to add pluggable support for such protocols. Accessed: 2026-08-25.

[34] Vitalik Buterin, Eric Conner, Rick Dudley, Matthew Slipper, Ian Norden, and Abdelhamid Bakhta. EIP-1559: Fee market change for ETH 1.0 chain. https://eips.ethereum.org/EIPS/eip-1559, 2019. Ethereum Improvement Proposal 1559, Standards Track: Core, Final; created April 13, 2019; accessed 2026-05-08.

[35] Vitalik Buterin, Diego Hernandez, Thor Kamphefner, Khiem Pham, Zhi Qiao, Danny Ryan, Juhyeok Sin, Ying Wang, and Yan X. Zhang. Combining GHOST and Casper. arXiv preprint arXiv:2003.03052, 2020.

[36] Vitalik Buterin, Jacob Illum, Matthias Nadler, Fabian Schär, and Ameen Soleimani. Blockchain privacy and regulatory compliance: Towards a practical equilibrium. *Blockchain: Research and Applications*, 5(1):100176, 2024. doi:10.1016/j.bcra.2023.100176.

[37] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. In *22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 668–679, 2015.

[38] CertiK. 2025 annual wrench attack report. https://www.certik.com/resources/blog, February 2026. 72 verified wrench attacks in 2025 (+75% year over year) and $40.9M in confirmed losses; reported by Decrypt and Cointelegraph, Feb. 2026. Accessed: 2026-06-06.

[39] Chainalysis. 2025 crypto crime mid-year update. https://www.chainalysis.com/blog/2025-crypto-crime-mid-year-update/, July 2025. Documents the correlation between Bitcoin price and physical attacks, and social-media and breach disclosure of holdings as a target-identification vector. Accessed: 2026-06-06.

[40] Chainalysis Team. Inside the KelpDAO bridge exploit: How approximately $292 million in rsETH was released against a non-existent burn. https://www.chainalysis.com/blog/kelpdao-bridge-exploit-april-2026/, 2026. Published April 23, 2026; accessed: 2026-05-09.

[41] ChainScore Labs. Why sequencer censorship is the next big l2 battlefield. https://chainscorelabs.com/blog/layer-2-wars-arbitrum-optimism-base-and-beyond/sequencer-economics-and-decentralization/why-sequencer-censorship-is-the-next-big-l2-battlefield, 2025. Accessed: 2026-05-09.

[42] Stefanos Chaliasos, Denis Firsov, and Benjamin Livshits. Towards a formal foundation for blockchain zk rollups. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pages 2714–2728, 2025.

[43] David Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, 1(1):65–75, 1988. doi:10.1007/BF00206326.

[44] David L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981. doi:10.1145/358549.358563.

[45] Weimin Chen and Xiapu Luo. MEVisor: High-throughput MEV discovery in DEXs with GPU parallelism. In *Network and Distributed System Security Symposium (NDSS)*, 2026.

[46] Tarun Chitra. Towards a theory of maximal extractable value II: Uncertainty. *arXiv preprint arXiv:2309.14201*, 2023.

[47] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private Information Retrieval. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 41–50, 1995. doi:10.1109/SFCS.1995.492461.

[48] CNN. Father of crypto entrepreneur rescued from kidnappers after having finger severed. https://www.cnn.com/2025/05/04/europe/crypto-entrepreneur-finger-severed-intl-latam, May 2025. Paris kidnapping; victim's son made his fortune in crypto. Accessed: 2026-06-03.

[49] Coinbase. Base node: Coinbase's managed RPC endpoint for base. https://www.coinbase.com/developer-platform/products/base-node, 2026. Accessed: 2026-08-25.

[50] CoinDesk. Amazon's AWS failure shakes up crypto's core promise. https://www.coindesk.com/news-analysis/2025/10/21/crypto-s-decentralized-illusion-shattered-again-by-another-aws-meltdown, October 2025. Reports Infura, Coinbase, Base, and Robinhood all disrupted by the October 20, 2025 AWS us-east-1 outage. Accessed: 2026-08-25.

[51] CoinDesk. Ledger co-founder's kidnapping sheds light on soaring crypto robberies. https://www.coindesk.com/policy/2025/01/24/ledger-co-founder-s-kidnapping-sheds-light-on-soaring-crypto-robberies, January 2025. David Balland abducted in France; finger severed and a multimillion-dollar Bitcoin ransom demanded. Accessed: 2026-06-03.

[52] CoinLaw. Rabby wallet usage statistics. https://coinlaw.io/rabby-wallet-statistics/, 2025. Accessed: 2026-05-09.

[53] CompareNodes. Top 211 providers for rpc nodes and blockchain apis in 2026. https://www.comparenodes.com/providers/, 2026. Accessed: 2026-08-24.

[54] Consensys. Consensys raises $450m series d funding as leading self-custodial wallet metamask reaches over 30 million maus. https://consensys.io/blog/consensys-raises-450m-series-d-funding, March 2022. Accessed: 2026-06-06.

[55] Consensys. MetaMask Reveals 55% Surge in Users, Introduces Default Security Alerts to Drive Wider Adoption And Prevent Billions Lost to Fraud. https://consensys.io/blog/metamask-reveals-55-surge-in-users-introduces-default-security-alerts-to, 2024. Published February 20, 2024; accessed 2026-05-07.

[56] A. Cooper, F. Gont, and D. Thaler. Security and privacy considerations for IPv6 address generation mechanisms. RFC 7721, IETF, 2016.

[57] Victor Costan and Srinivas Devadas. Intel SGX explained. *IACR Cryptology ePrint Archive*, 2016/086, 2016.

[58] CoW DAO. MEV Blocker. https://mevblocker.io/, 2026. Accessed: 2026-05-09.

[59] Davide Crapis and Vitalik Buterin. ZK API usage credits: LLMs and beyond. Ethereum Research (ethresear.ch), 2026. URL: https://ethresear.ch/t/zk-api-usage-credits-llms-and-beyond/24104.

[60] CryptoBriefing. Infura, Alchemy block Tornado Cash following Treasury ban. https://cryptobriefing.com/infura-alchemy-block-tornado-cash-following-treasury-ban/, August 2022.

[61] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2020.

[62] L. Datskoluo and T. Craig. Railgun Hits Record $4bn in Volume as Privacy Demand Surges. https://www.dlnews.com/articles/defi/railgun-sees-record-shielded-transaction-volume/, 2025. Published October 21, 2025; accessed 2026-05-07.

[63] Emma Dauterman, Vivian Fang, Ioannis Demertzis, Natacha Crooks, and Raluca Ada Popa. Snoopy: Surpassing the scalability bottleneck of oblivious storage. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles (SOSP)*, pages 655–671, 2021. Reports 18 SGX-enabled machines sustaining 92K requests/s at sub-500 ms latency over two million 160-byte objects.

[64] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *Proceedings on Privacy Enhancing Technologies*, 2018(3):164–180, 2018.

[65] Killian Davitt, Dan Ristea, and Steven J. Murdoch. Are we collaborative yet? a usability perspective on mixnet latency for real-time applications. arXiv preprint arXiv:2601.17845, 2026. URL: https://arxiv.org/abs/2601.17845.

[66] Christian Decker and Roger Wattenhofer. Information Propagation in the Bitcoin Network. In *Proceedings of the 13th IEEE International Conference on Peer-to-Peer Computing*, pages 1–10. IEEE, 2013. doi:10.1109/P2P.2013.6688704.

[67] Ioannis Demertzis, Dimitrios Papadopoulos, Charalampos Papamanthou, and Saurabh Shintre. SEAL: Attack mitigation for encrypted databases via adjustable leakage. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 2433–2450, 2020.

[68] Dominic Deuber, Viktoria Ronge, and Christian Rückert. SoK: Assumptions underlying cryptocurrency deanonymizations. *Proceedings on Privacy Enhancing Technologies*, 2022(3):670–691, 2022. doi:10.56553/popets-2022-0091.

[69] Claudia Diaz, Harry Halpin, and Aggelos Kiayias. The Nym Network: The Next Generation of Privacy Infrastructure, 2021. Whitepaper, Version 1.0.

[70] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, pages 303–320. USENIX Association, 2004.

[71] Jason A. Donenfeld. WireGuard: Next Generation Kernel Network Tunnel. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2017. doi:10.14722/NDSS.2017.23160.

[72] Nico Döttling, Lucjan Hanzlik, Bernardo Magri, and Stella Wohnig. McFly: Verifiable encryption to the future made practical. In *Financial Cryptography and Data Security*, pages 252–269. Springer, 2024. URL: https://eprint.iacr.org/2022/433, doi:10.1007/978-3-031-47754-6_15.

[73] Electronic Frontier Foundation. Telegram, user data, and government compulsion. https://www.eff.org/, 2025. Accessed: 2026-05-09.

[74] Espresso Systems. Shared sequencing documentation. https://docs.espressosys.com/, 2025. Accessed: 2026-05-09.

[75] Ethereum Community. EIP-8105: Universal enshrined encrypted mempool. https://eips.ethereum.org/EIPS/eip-8105, 2025. Ethereum Improvement Proposal, Draft. Accessed: 2026-05-09.

[76] Ethereum Foundation. Kohaku roadmap. https://notes.ethereum.org/@niard/KohakuRoadmap, 2025. Accessed: 2026-08-25.

[77] Ethereum.org. Spin up Your Own Ethereum Node. https://ethereum.org/developers/docs/nodes-and-clients/run-a-node/, 2026. Accessed: 2026-05-05.

[78] Rex Fernando, Guru-Vamsi Policharla, Andrei Tonkikh, and Zhuolun Xiang. TrX: Encrypted mempools in high performance BFT protocols. Cryptology ePrint Archive, 2025. URL: https://eprint.iacr.org/2025/1542.

[79] Christof Ferreira Torres, Fiona Willi, and Shweta Shinde. Is your wallet snitching on you? an analysis on the privacy implications of Web3. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 769–786. USENIX Association, 2023. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/torres.

[80] Flashbots. Flashbots MEV-Boost Block Builder. https://github.com/flashbots/builder, 2024. Archived repository; documents that the greedy block-building algorithm sorts bundles and transactions by effective gas price; accessed 2026-05-08.

[81] Flashbots. JSON-RPC Endpoints. https://docs.flashbots.net/flashbots-auction/advanced/rpc-endpoint, 2025. Documents Flashbots bundle RPC methods including `eth_sendBundle` and `eth_callBundle`; accessed 2026-05-08.

[82] Flashbots. Buildernet: Decentralized block building with trusted execution environments. https://buildernet.org/, 2026. Accessed: 2026-05-09.

[83] Flashbots. Flashbots protect documentation. https://docs.flashbots.net/flashbots-protect/overview, 2026.

[84] Flashbots. Mev-boost documentation. https://docs.flashbots.net/flashbots-mev-boost/introduction, 2026.

[85] Flashbots. Network-anonymized mempools and flashnet. https://writings.flashbots.net/network-anonymized-mempools, 2026. Accessed: 2026-05-09.

[86] Flashbots. Sending tx and bundles. https://docs.flashbots.net/guide-send-tx-bundle, 2026. Accessed: 2026-08-24.

[87] Flashbots Research and Engineering Team. SUAVE-Geth: Single unifying auction for value expression. https://github.com/flashbots/suave-geth, 2023. Accessed: July 20, 2025.

[88] Fluidkey. Fluidkey stealth address implementation and metrics. https://docs.fluidkey.com/, 2026. Accessed: 2026-05-09.

[89] Forbes. Forbes Releases First-Ever Crypto Rich List, A Compilation Of The 20 Wealthiest People In Crypto. https://www.forbes.com/sites/forbespr/2018/02/07/forbes-releases-first-ever-crypto-rich-list-a-compilation-of-the-20-wealthiest-people-in-crypto, 2018. Published February 7, 2018; accessed 2026-05-09.

[90] Emilio Frangella and Lasse Herskind. Aave V3 Technical Paper. Technical paper, Aave, January 2022. Available: https://raw.githubusercontent.com/aave/aave-v3-core/master/techpaper/Aave_V3_Technical_Paper.pdf. Accessed: 2026-05-05.

[91] Kyle Fredrickson, Ioannis Demertzis, James P Hughes, and Darrell DE Long. Sparta: Practical anonymity with long-term resistance to traffic analysis. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 1457–1473. IEEE, 2025.

[92] Sergey Frolov and Eric Wustrow. The use of TLS in censorship circumvention. In *26th Annual Network and Distributed System Security Symposium (NDSS 2019)*. The Internet Society, 2019.

[93] Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert van Renesse, and Emin Gün Sirer. Decentralization in Bitcoin and Ethereum Networks. In *Financial Cryptography and Data Security*, pages 439–457. Springer, 2018. doi:10.1007/978-3-662-58387-6_24.

[94] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 169–178, 2009. doi:10.1145/1536414.1536440.

[95] Oded Goldreich and Rafail Ostrovsky. Software Protection and Simulation on Oblivious RAMs. *Journal of the ACM*, 43(3):431–473, 1996. doi:10.1145/233551.233553.

[96] Lauren Goode. Coinbase says its data breach affects at least 69,000 customers. https://techcrunch.com/2025/05/21/coinbase-says-its-data-breach-affects-at-least-69000-customers/, May 2025. Accessed: 2026-05-09.

[97] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. SoK: Layer-Two Blockchain Protocols. In *Financial Cryptography and Data Security*, pages 201–226. Springer, 2020. doi:10.1007/978-3-030-51280-4_12.

[98] Tina Gupta, Mallesh M. Pai, and Max Resnick. The centralizing effects of private order flow on proposer-builder separation. In *Advances in Financial Technologies (AFT)*, 2023.

[99] Lioba Heimbach, Lucianna Kiffer, Christof Ferreira Torres, and Roger Wattenhofer. Ethereum's Proposer-Builder Separation: Promises and Realities. In *Proceedings of the ACM Internet Measurement Conference*, pages 406–420, 2023. doi:10.1145/3618257.3624824.

[100] Lioba Heimbach, Yann Vonlanthen, Juan Villacis, Lucianna Kiffer, and Roger Wattenhofer. Deanonymizing Ethereum validators: The P2P network has a privacy issue. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 1319–1338. USENIX Association, 2025. URL: https://www.usenix.org/conference/usenixsecurity25/presentation/heimbach.

[101] Lioba Heimbach and Roger Wattenhofer. SoK: Preventing transaction reordering manipulations in decentralized finance. In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, AFT '22, pages 47–60, 2022. doi:10.1145/3558535.3559784.

[102] Ryan Henry, Amir Herzberg, and Aniket Kate. Blockchain access privacy: Challenges and directions. *IEEE Security & Privacy*, 16(4):38–45, 2018. doi:10.1109/MSP.2018.3111245.

[103] Alexandra Henzinger, Matthew M. Hong, Henry Corrigan-Gibbs, Sarah Meiklejohn, and Vinod Vaikuntanathan. One server for the price of two: Simple and fast single-server private information retrieval. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 3889–3905. USENIX Association, 2023. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/henzinger.

[104] Everett Hildenbrandt, Manasvi Saxena, Xiaoran Zhu, Nishant Rodrigues, Philip Daian, Dwight Guth, Brandon Moore, Daejun Park, Yi Zhang, Andrei Stefanescu, and Grigore Rosu. KEVM: A Complete Formal Semantics of the Ethereum Virtual Machine. In *Proceedings of the IEEE Computer Security Foundations Symposium (CSF)*, pages 204–217, 2018. doi:10.1109/CSF.2018.00022.

[105] Hinkal. 24 Confidential Transaction Statistics 2026. https://hinkal.io/blog/confidential-transaction-statistics, 2026. Accessed 2026-05-07.

[106] HOPR. RPCh documentation and deployment status. https://docs.hoprnet.org/developers/rpch, 2024. Accessed: 2026-05-09.

[107] IBM Security Intelligence. 83% of organizations reported insider attacks in 2024. https://securityintelligence.com/articles/83-percent-organizations-reported-insider-threats-2024/, February 2025. Accessed: 2026-05-09.

[108] imToken. Scam warning: Fraudulent RPC node adjustments. https://support.token.im/hc/en-us/articles/31423984023961-Scam-Warning-Fraudulent-RPC-Node-Adjustments, 2026. Updated January 23, 2026; accessed: 2026-05-07.

[109] Infura. Decentralized infrastructure network gateway documentation. https://docs.metamask.io/services/concepts/infura-din/, 2026. Accessed: 2026-05-09.

[110] Investing.com. Ankr is now the leading web3 rpc provider with 6b requests daily. https://www.investing.com/news/cryptocurrency-news/ankr-is-now-the-leading-web3-rpc-provider-with-6b-requests-daily-2809244, April 2022. Accessed: 2026-06-07.

[111] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *19th Annual Network and Distributed System Security Symposium (NDSS)*, 2012.

[112] IVPN. When law enforcement knocks on a VPN's door: What happens? https://www.ivpn.net/blog/when-law-enforcement-knocks-on-a-vpns-door-what-happens/, 2024. Accessed: 2026-05-09.

[113] Emma Jagger. Ip geolocation accuracy: How reliable is it? AbstractAPI. https://www.abstractapi.com/guides/ip-geolocation/how-accurate-is-ip-geolocation, 2025. Accessed: 2026-05-09.

[114] Paul Janicot and Alex Vinyas. Private mev protection rpcs: Benchmark study. *arXiv preprint arXiv:2505.19708*, 2025.

[115] Simon Jentzsch and Christoph Jentzsch. EIP-1186: RPC-Method to Get Merkle Proofs – eth_getProof. https://eips.ethereum.org/EIPS/eip-1186, 2018. Accessed: 2026-05-05.

[116] jmc. State of light clients, and how Portal can help. https://blog.ethportal.net/posts/light-clients, February 2023. Ethereum Portal Network blog, published February 1, 2023.

[117] JSON-RPC Working Group. Json-rpc 2.0 specification. https://www.jsonrpc.org/specification, 2010.

[118] Kate Keahey, Pierre Riteau, Dan Stanzione, Tim Cockerill, Joe Mambretti, Paul Rad, and Paul Ruth. Chameleon: A scalable production testbed for computer science research. In Jeffrey Vetter, editor, *Contemporary High Performance Computing: From Petascale toward Exascale, Volume 3*, chapter 5, pages 123–148. Chapman & Hall/CRC Computational Science, 2019.

[119] Mahimna Kelkar, Fan Zhang, Steven Goldfeder, and Ari Juels. Order-fairness for Byzantine consensus. In *Advances in Cryptology – CRYPTO 2020*, volume 12172 of *Lecture Notes in Computer Science*, pages 451–480. Springer, 2020. doi:10.1007/978-3-030-56877-1_16.

[120] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. Generic attacks on secure outsourced databases. In *23rd ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1329–1340, 2016.

[121] Shea Ketsdever. 2 million protect users. https://writings.flashbots.net/2m-protect-users, 2024. Published October 30, 2024; accessed 2026-05-07.

[122] Amirhossein Khajehpour, Hanzaleh Akbarinodehi, Mohammad Jahanara, and Chen Feng. A mempool encryption scheme for ethereum via multiparty delay encryption. *Cryptology ePrint Archive*, 2023.

[123] Amit Klein and Benny Pinkas. From IP ID to device ID and KASLR bypass. 28th USENIX Security Symposium (USENIX Security 19), 2019.

[124] Kohaku. Kohaku privacy sdk roadmap. https://github.com/ethereum/kohaku, 2025. Accessed: 2026-05-09.

[125] Ivan Krstić. Keynote: The evolution of Apple Private Cloud Compute. Confidential Computing Summit 2026; https://www.youtube.com/watch?v=-kKpTcbhUBk, 2026. Accessed: 2026-08-24.

[126] Lava Network. Lava protocol overview. https://docs.lavanet.xyz/lava-architecture/, 2026. Documents that the consumer connects directly to the chosen RPC node provider, which executes the call against its own full-node infrastructure. Accessed: 2026-08-25.

[127] LayerZero Labs. LayerZero labs KelpDAO incident report. https://layerzero.network/blog/layerzero-labs-kelpdao-incident-report, 2026. Published May 20, 2026 (companion PDF dated May 18, 2026: https://layerzero.network/publications/kelpdao-incident-report.pdf); describes the attacker poisoning internal RPC nodes to return tampered responses to the Decentralized Verifier Network while returning correct responses to LayerZero's own monitoring tools querying the same nodes, releasing rsETH against a non-existent burn. Accessed: 2026-08-24.

[128] Le Monde. Telegram gave more user data to french authorities after founder's arrest. https://www.lemonde.fr/en/pixels/article/2025/01/08/telegram-gave-more-user-data-to-french-authorities-after-founder-s-arrest_6736824_13.html, January 2025. Accessed: 2026-05-09.

[129] Chuanlei Li, Zhicheng Sun, Jing Xin Yuu, and Xuechao Wang. The walls have ears: Unveiling cross-chain sandwich attacks in DeFi. https://arxiv.org/abs/2511.15245, 2025. arXiv:2511.15245.

[130] Kai Li, Jiaqi Chen, Xianghong Liu, Yuzhe Tang, XiaoFeng Wang, and Xiapu Luo. As Strong As Its Weakest Link: How to Break Blockchain DApps at RPC Service. In *Network and Distributed System Security Symposium (NDSS)*, 2021. doi:10.14722/ndss.2021.23108.

[131] Kai Li, Yuzhe Tang, Jiaqi Chen, Yibo Wang, and Xianghong Liu. TopoShot: Uncovering Ethereum's Network Topology Leveraging Replacement Transactions. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2021. doi:10.1145/3487552.3487814.

[132] Kai Li, Yibo Wang, and Yuzhe Tang. DETER: Denial of Ethereum Txpool sERvices. In *Proceedings of*

the ACM SIGSAC Conference on Computer and Communications Security (CCS), pages 1645–1667, 2021. doi:10.1145/3460120.3485369.

[133] Zhen Li, Yi Zheng, Qi Li, Ming Wu, and Kunbin Peng. Dragnet: A method for tagging Bitcoin addresses of exchanges. TechRxiv preprint, 2020.

[134] Dan Lin, Jiajing Wu, Tao Huang, Kaixin Lin, and Zibin Zheng. Who is who on Ethereum? account labeling using heterophilic graph convolutional network. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 54(3):1541–1553, 2024. doi:10.1109/TSMC.2023.3329381.

[135] Yehuda Lindell. Secure multiparty computation. *Communications of the ACM*, 64(1):86–96, 2020. doi:10.1145/3387108.

[136] Ken Ziyu Liu and Erik Chi. Unlinkable inference as a user privacy architecture. The Open Anonymity Project Blog, February 2026. Accessed: 2026-05-28. URL: https://openanonymity.ai/blog/unlinkable-inference/.

[137] Angelique Faye Loe, Liam Medley, Christian O'Connell, and Elizabeth Quaglia. A practical verifiable delay function and delay encryption scheme. Cryptology ePrint Archive, 2021.

[138] F. Luo, Zihao Li, W. Luo, Z. He, Xiapu Luo, Z. Ma, S. Song, and T. Chen. Light into darkness: Demystifying profit strategies throughout the MEV bot lifecycle. In *Network and Distributed System Security Symposium (NDSS)*, 2026.

[139] Zhongtang Luo, Rohan Murukutla, and Aniket Kate. Last Mile of Blockchains: RPC and Node-as-a-Service. arXiv preprint arXiv:2212.03383, 2022.

[140] Yuval Marcus, Ethan Heilman, and Sharon Goldberg. Low-Resource Eclipse Attacks on Ethereum's Peer-to-Peer Network. Cryptology ePrint Archive, Paper 2018/236, 2018. URL: https://eprint.iacr.org/2018/236.

[141] Sinisa Matetic, Karl Wüst, Moritz Schneider, Kari Kostiainen, Ghassan Karame, and Srdjan Capkun. BITE: Bitcoin Lightweight Client Privacy using Trusted Execution. In *Proceedings of the USENIX Security Symposium*, pages 783–800, 2019.

[142] MaxMind. Geolocation accuracy. MaxMind Support Center. https://support.maxmind.com/hc/en-us/articles/4407630607131-Geolocation-Accuracy, 2025. Accessed: 2026-05-09.

[143] K. G. Orphanides McDonald. IPVanish Review: A Quick and Inexpensive VPN with a Shady Past. https://www.wired.com/story/ipvanish-vpn-review, August 2019. Accessed: 2026-05-06.

[144] Jesse McKinley. Crypto investor tortured in manhattan kidnapping scheme. https://www.nytimes.com/2025/05/24/nyregion/crypto-investor-torture-italian-tourist.html, May 2025. Accessed: 2026-05-09.

[145] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A fistful of bitcoins: Characterizing payments among men with no names. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2013.

[146] Samir Jordan Menon and David J. Wu. SPIRAL: Fast, high-rate single-server PIR via FHE composition. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 930–947. IEEE, 2022. URL: https://eprint.iacr.org/2022/368.

[147] MetaMask. eth_sendTransaction | MetaMask developer documentation. https://docs.metamask.io/wallet/reference/eth_sendtransaction/, 2026. Documents the wallet confirmation prompt shown to the user before submission. Accessed: 2026-08-24.

[148] MetaMask. MetaMask core: Shared logic and services for MetaMask client codebases. https://github.com/MetaMask/core, 2026. Balance polling in `AccountTrackerController.ts` (`eth_getBalance`) and `multicall.ts` (`balanceOf` via `eth_call`); nonce via `@metamask/nonce-tracker` (`eth_getTransactionCount`); fallback fee estimation in `gas-fee-controller` (`eth_gasPrice`); gas estimation, submission, and receipt polling in `transaction-controller` (`eth_estimateGas`, `eth_sendRawTransaction`, `eth_getTransactionReceipt`). Accessed: 2026-08-24.

[149] MetaMask Help Center. What is Infura, and Why Does MetaMask Use It? https://support.metamask.io/more-web3/learn/what-is-infura-and-why-does-metamask-use-it/, 2026. Accessed: 2026-05-05.

[150] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *2013 IEEE Symposium on Security and Privacy*, pages 397–411, 2013. doi:10.1109/SP.2013.34.

[151] Marija Mikić, Mihajlo Srbakoski, and Strahinja Praška. Post-quantum stealth address protocols. Cryptology ePrint Archive, Paper 2025/112, 2025.

[152] Mysterium Network. Mysterium network documentation and node metrics. https://docs.mysterium.network/, 2026. Accessed: 2026-05-09.

[153] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf, 2008.

[154] Ray Neiheiser, Gustavo Inacio, Luciana Rech, Carlos Montez, Miguel Matos, and Luis Rodrigues. Practical Limitations of Ethereum's Layer-2. *IEEE Access*, 11:8651–8662, 2023. doi:10.1109/ACCESS.2023.3237897.

[155] Danny Nelson. Crypto industry's sanctions woes on full display in MetaMask's Venezuela hiccup. https://web.archive.org/web/20220516095735/https://www.coindesk.com/business/2022/03/04/crypto-industrys-sanctions-woes-on-full-display-in-metamasks-venezuela-hiccup/, March 2022. CoinDesk; archived March 4, 2022. Accessed 2026-05-09.

[156] Nym Technologies. Nym mixnet documentation and integrations. https://docs.nym.com/, 2026. Accessed: 2026-05-09.

[157] Oasis Protocol. Oasis sapphire confidential smart contract platform. https://oasisprotocol.org/sapphire, 2026. Accessed: 2026-05-09.

[158] Oblivious Labs. TEE+ORAM wbtc demonstration. https://obliviouslabs.com/WBTCdemo/, 2026. Accessed: 2026-08-25.

[159] OKX. X layer developer documentation: Sequencer and RPC endpoints. https://web3.okx.com/xlayer/docs/developer/build-on-xlayer/about-xlayer, 2026. Accessed: 2026-08-25.

[160] Orchid. Orchid decentralized vpn documentation. https://docs.orchid.com/, 2026. Accessed: 2026-05-09.

[161] Marilyne Ordekian, Gilberto Atondo-Siu, Alice Hutchings, and Marie Vasek. Investigating wrench attacks: Physical attacks targeting cryptocurrency users. In *6th Conference on Advances in Financial Technologies (AFT 2024)*, volume 316 of *LIPIcs*, pages 24:1–24:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. Only peer-reviewed study to date that systematically defines and quantifies wrench attacks. Accessed: 2026-06-06. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.AFT.2024.24, doi:10.4230/LIPIcs.AFT.2024.24.

[162] Burak Öz, Danning Sui, Thomas Thiery, and Florian Matthes. Who wins ethereum block building auctions and why? In *6th Conference on Advances in Financial Technologies (AFT 2024)*, volume 316 of *LIPIcs*, pages 22:1–22:25, 2024. doi:10.4230/LIPIcs.AFT.2024.22.

[163] Daniel Pérez, Sam M. Werner, Jiahua Xu, and Benjamin Livshits. Liquidations: Defi on a knife-edge. In *Financial Cryptography and Data Security*, 2021.

[164] Phantom. Phantom series c and user metrics announcement. https://phantom.com/learn/blog/series-c, 2025. Accessed: 2026-05-09.

[165] Ania M. Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The Loopix Anonymity System. In *26th USENIX Security Symposium*, pages 1199–1216, 2017.

[166] Pocket Network. pocket-relay-miner: Protocol specification. https://github.com/pokt-network/pocket-relay-miner/blob/main/docs/PROTOCOL_SPEC.md, 2026. Documents the relay payload as `raw_json_rpc`, signed but not encrypted, forwarded in the clear to the backend node. Accessed: 2026-08-25.

[167] Joseph Poon and Thaddeus Dryja. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. Draft Version 0.5.9.2, 2016. URL: https://lightning.network/lightning-network-paper.pdf.

[168] Privacy Stewards of Ethereum. PSE roadmap: 2025 and beyond. https://pse.dev/blog/pse-roadmap-2025, 2025. Focus areas: private writes, private reads, and private proving. Accessed: 2026-08-24.

[169] Yuxin Qi, Jun Wu, Hansong Xu, and Mohsen Guizani. Blockchain data mining with graph learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(2):729–748, 2024. doi:10.1109/TPAMI.2023.3326031.

[170] Kaihua Qin, Henryk Hadass, Arthur Gervais, and Joel Reardon. Applying Private Information Retrieval to Lightweight Bitcoin Clients. In *Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 60–72, 2019. Extended version: arXiv:2008.11358. doi:10.1109/CVCBT.2019.00012.

[171] Kaihua Qin, Liyi Zhou, Benjamin Livshits, and Arthur Gervais. Quantifying blockchain extractable value: How dark is the forest? In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 198–214. IEEE, 2022.

[172] Fergal Reid and Martin Harrigan. An analysis of anonymity in the Bitcoin system. In Yaniv Altshuler, Yuval Elovici, Armin B. Cremers, Nadav Aharony, and Alex Pentland, editors, *Security and Privacy in Social Networks*, pages 197–223. Springer, New York, NY, 2013. doi:10.1007/978-1-4614-4139-7_10.

[173] Eric Rescorla. The transport layer security (tls) protocol version 1.3. Technical Report RFC 8446, IETF, 2018.

[174] Reuters. How Reuters Tallied the Trump Organization's Crypto Income. https://www.reuters.com/investigations/how-reuters-tallied-trump-organizations-crypto-income-2025-10-28/, 2025. Published October 28, 2025; accessed 2026-05-09.

[175] Francisco Rodrigues. Arkham Launches New Tag to Track Crypto Influencers' Wallets. https://www.coindesk.com/markets/2025/03/08/arkham-launches-new-tag-to-track-crypto-influencers-wallets, 2025. Published March 8, 2025; accessed 2026-05-09.

[176] Nicolò Romandini, Carlo Mazzocca, Kai Otsuki, and Rebecca Montanari. SoK: Security and privacy of AI agents for blockchain. In *2025 7th International Conference on Blockchain Computing and Applications (BCCA)*, pages 708–720. IEEE, 2025. doi:10.1109/BCCA66705.2025.11229689.

[177] Michael Rosenberg, Maurice Shih, Zhenyu Zhao, Rui Wang, Ian Miers, and Fan Zhang. ZIPNet: Low-bandwidth anonymous broadcast from (dis)trusted execution environments. In *Proceedings on Privacy Enhancing Technologies Symposium (PETS)*, pages 211–225, 2025. URL: https://eprint.iacr.org/2024/1227.

[178] Tim Roughgarden. Foundations of Blockchain Protocols. https://timroughgarden.org/notes.html, 2021. Lecture notes for COMS 6998, Columbia University. Accessed: 2026-05-05.

[179] Samuel Schlesinger and Jonathan Katz. Anonymous credit tokens. Internet-Draft draft-schlesinger-cfrg-act-01, IETF, 2026. Work in Progress.

[180] Secret Network Team. Secret Network: Graypaper. https://scrt.network/graypaper, 2020. Accessed: July 20, 2025.

[181] Secret Network Team. Secret network graypaper. https://scrt.network/graypaper, 2020. Accessed: 2026-05-09.

[182] Security.org. Annual vpn consumer report. https://www.security.org/resources/vpn-consumer-report-annual/, 2025. Accessed: 2026-05-09.

[183] Sentinel. Sentinel dvpn documentation. https://docs.sentinel.co/, 2026. Accessed: 2026-05-09.

[184] Shutter Network. encrypting-rpc-server: An Ethereum JSON RPC server for shutterized Gnosis chain that encrypts transactions. https://github.com/shutter-network/encrypting-rpc-server, 2026. Documents that clients may send plaintext transactions to this trusted RPC endpoint, which performs the encryption on their behalf. Accessed: 2026-08-25.

[185] Storm Slivkoff and Georgios Konstantopoulos. How to raise the gas limit, part 1: State growth. https://www.paradigm.xyz/2024/03/how-to-raise-the-gas-limit-1, 2024. Paradigm; published March 4, 2024.

[186] SlowMist. Unveiling a new scam: Malicious modification of RPC node links to steal assets. https://slowmist.medium.com/unveiling-a-new-scam-malicious-modification-of-rpc-node-links-to-steal-assets-2ca324200853, 2024. Published April 25, 2024; accessed: 2026-05-09.

[187] Solidity Authors. Contract ABI Specification. https://docs.soliditylang.org/en/latest/abi-spec.html.

[188] Frank Stewskid. Aave november 2023 security incident: Bug bounty report and response. https://defiteller.com/aave-protocol-security-update-november-2023, November 2023. Accessed: 2026-05-09.

[189] Erik Sy, Christian Burkert, Hannes Federrath, and Mathias Fischer. Tracking users across the web via TLS session resumption. In *Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC '18)*, pages 289–299. ACM, 2018. doi:10.1145/3274694.3274708.

[190] Yuhang Tang, Chunpeng Xu, Chi Zhang, Yuncheng Wu, and Liehuang Zhu. Analysis of address linkability in tornado cash on ethereum. In *Cyber Security*, Communications in Computer and Information Science, pages 39–50. 2022. doi:10.1007/978-981-16-9229-1_3.

[191] TEN Protocol. TEN protocol confidential l2 documentation. https://docs.ten.xyz/, 2026. Accessed: 2026-05-09.

[192] Tendermint Core. Light client. https://docs.tendermint.com/v0.34/tendermint-core/light-client.html, 2021. Tendermint Core documentation, v0.34.

[193] The Defiant. Infura expands decentralized infra network to EigenLayer following AWS outage. https://thedefiant.io/news/infrastructure/consensys-infura-expands-din-to-eigenlayer, November 2025. Accessed: 2026-05-19.

[194] The Graph. Graph node. https://thegraph.com/docs/en/indexing/tooling/graph-node/, 2026. Documents provider-capability requirements (`archive`, `traces`/`trace_filter`) for indexing subgraphs. Accessed: 2026-08-24.

[195] The Register. VPN Logs Helped Unmask Alleged 'Net Stalker, Say Feds. https://www.theregister.com/2017/10/08/vpn_logs_helped_unmask_alleged_net_stalker_say_feds/, October 2017. Accessed: 2026-05-06.

[196] The Tor Project. Tor Metrics: OnionPerf Latencies. https://metrics.torproject.org/onionperf-latencies.html. Continuously updated measurements of Tor request latency. Accessed: 2026-06-06.

[197] Martin Thomson, Tommy Pauly, and David Schinazi. Oblivious HTTP. Technical Report RFC 9458, IETF, 2024. doi:10.17487/RFC9458.

[198] TokenPocket. Fake RPC nodes, fake balances. https://help.tpwallet.io/en/security-knowledge/common-fraud-cases/fake-rpc-nodes-fake-balances, 2025. Accessed: 2026-05-07.

[199] Tor Project. Tor documentation and usage metrics. https://metrics.torproject.org/, 2026. Accessed: 2026-05-09.

[200] Christof Ferreira Torres, Ramiro Camino, and Radu State. Frontrunner Jones and the raiders of the dark forest: An empirical study of frontrunning on the Ethereum blockchain. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1343–1359. USENIX Association, 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/torres.

[201] Trireme Research. Shared sequencing research notes. https://trireme.io/, 2025. Accessed: 2026-05-09.

[202] Truffle Suite. Ganache: A personal blockchain for Ethereum development. https://github.com/trufflesuite/ganache, 2026. Accessed: 2026-08-25.

[203] U.S. Department of Justice. Court Authorizes Service of John Doe Summons Seeking the Identities of U.S. Taxpayers Who Have Used Virtual Currency. https://www.justice.gov/archives/opa/pr/court-authorizes-service-john-doe-summons-seeking-identities-us-taxpayers-who-have-used, 2016. Published November 30, 2016; accessed 2026-05-09.

[204] U.S. Department of Justice. Court Authorizes Service of John Doe Summons Seeking Identities of U.S. Taxpayers Who Have Used Cryptocurrency. https://www.justice.gov/opa/pr/court-authorizes-service-john-doe-summons-seeking-identities-us-taxpayers-who-have-used-1, 2021. Published May 5, 2021; accessed 2026-05-09.

[205] U.S. Department of the Treasury. U.S. Treasury Sanctions Notorious Virtual Currency Mixer Tornado Cash. https://home.treasury.gov/news/press-releases/jy0916, 2022. Published August 8, 2022; accessed 2026-05-07.

[206] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 991–1008. USENIX Association, 2018. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/bulck.

[207] Stephan Van Schaik, Andrew Kwong, Daniel Genkin, and Yuval Yarom. SGAxe: How SGX fails in practice. https://sgaxe.com/files/SGAxe.pdf, 2020. Self-published technical report / attack disclosure, not peer-reviewed.

[208] Friedhelm Victor. Address clustering heuristics for Ethereum. In *Financial Cryptography and Data Security Workshops*, pages 617–633. Springer, 2020. doi:10.1007/978-3-030-54455-3_44.

[209] Anton Wahrstätter, Jens Ernstberger, Aviv Yaish, Liyi Zhou, Kaihua Qin, Taro Tsuchiya, Sebastian Steinhorst, Davor Svetinovic, Nicolas Christin, Mikolaj Barczentewicz, and Arthur Gervais. Blockchain censorship. In *Proceedings of the ACM Web Conference 2024*, WWW '24, page 1632–1643, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3589334.3645431.

[210] Anton Wahrstätter, Matthew Solomon, Ben DiFrancesco, and Vitalik Buterin. ERC-5564: Stealth Addresses. https://eips.ethereum.org/EIPS/eip-5564, 2022. Accessed: 2026-05-05.

[211] Anton Wahrstätter, Matthew Solomon, Ben DiFrancesco, and Vitalik Buterin. ERC-6538: Stealth Meta-Address Registry. https://eips.ethereum.org/EIPS/eip-6538, 2023. Accessed: 2026-05-05.

[212] Anton Wahrstätter, Matthew Solomon, Ben DiFrancesco, Vitalik Buterin, and Davor Svetinovic. BaseSAP: Modular Stealth Address Protocol for Programmable Blockchains. *IEEE Transactions on Information Forensics and Security*, 19:3539–3553, 2024. doi:10.1109/TIFS.2024.3364081.

[213] Ruida Wang, Ziyi Li, Xianhui Lu, Zhenfei Zhang, and Kunpeng Wang. Key derivable signature and its application in blockchain stealth address. *Cybersecurity*, 7(43), 2024. doi:10.1186/s42400-024-00231-x.

[214] Shan Wang, Ming Yang, Yu Liu, Yue Zhang, Shuaiqing Zhang, Zhen Ling, Jiannong Cao, and Xinwen Fu. Time Tells All: Deanonymization of Blockchain RPC Users with Zero Transaction Fee. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 3490–3504, 2025. doi:10.1145/3719027.3765082.

[215] Weihong Wang, Yana Dimova, Victor Vansteenkiste, Tom Van Goethem, and Tom Van Cutsem. The masks we (think we) wear: Privacy threats of browser-extension wallets in the Web3 ecosystem. *Proceedings on Privacy Enhancing Technologies*, 2026(3), 2026.

[216] Weihong Wang and Tom Van Cutsem. Depermissioning web3: A permissionless accountable RPC protocol for blockchain networks. In *45th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2025. Reports Infura at 47.52% and Alchemy at 31.07% of measured dApp RPC connections. arXiv:2506.03940.

[217] Yibo Wang, Yuzhe Tang, Kai Li, Wanning Ding, and Zhihua Yang. Understanding Ethereum Mempool Security under Asymmetric DoS by Symbolized Stateful Fuzzing. In *Proceedings of the USENIX Security Symposium*, 2024.

[218] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I. Weidele, Claudio Bellei, Tom Robinson, and Charles E. Leiserson. Anti-money laundering in Bitcoin: Experimenting with graph convolutional networks for financial forensics. arXiv preprint arXiv:1908.02591, 2019.

[219] Philipp Winter, Richard Köwer, Martin Mulazzani, Markus Huber, Sebastian Schrittwieser, Stefan Lindskog, and Edgar Weippl. Spoiled onions: Exposing malicious tor exit relays. In *Privacy Enhancing Technologies Symposium (PETS)*, pages 304–331, 2014. URL: https://arxiv.org/abs/1401.4917.

[220] Gavin Wood. Ethereum: A Secure Decentralised Generalised Transaction Ledger. Ethereum Yellow Paper, 2014. Available at https://ethereum.github.io/yellowpaper/paper.pdf.

[221] Fei Wu, Danning Sui, Thomas Thiery, and Mallesh Pai. Measuring CEX-DEX extracted value and searcher profitability: The darkest of the MEV dark forest. In *7th Conference on Advances in Financial Technologies (AFT 2025)*, 2025. URL: https://arxiv.org/abs/2507.13023.

[222] Zhiying Wu, Jieli Liu, Jiajing Wu, and Zibin Zheng. TRacer: Scalable graph-based transaction tracing for account-based blockchain trading systems. *IEEE Transactions on Information Forensics and Security*, 18:2609–2621, 2023. doi:10.1109/TIFS.2023.3266162.

[223] Gengsheng Xue, Yang Li, and Zibin Zheng. Transparency to the extreme: An in-depth study of the Bitcoin exchange ecosystem. In *3rd International Conference on Blockchain and Trustworthy Systems (BlockSys)*, pages 257–271, 2021.

[224] Sen Yang, Fan Zhang, Ken Huang, Xi Chen, Youwei Yang, and Feng Zhu. Sok: Mev countermeasures: Theory and practice. *arXiv preprint arXiv:2212.05111*, 2022.

[225] Sen Yang, Fan Zhang, Ken Huang, Xi Chen, Youwei Yang, and Feng Zhu. SoK: MEV countermeasures. In *Proceedings of the Workshop on Decentralized Finance and Security*, pages 21–30, 2024.

[226] Yifang Zhang, Mingyue Wang, Fangda Guo, Xidi Qu, Yu Guo, and Shengling Wang. OblivChain: Enabling oblivious queries for blockchain light clients with malicious security. In *Database Systems for Advanced Applications (DASFAA 2024)*, Lecture Notes in Computer Science, pages 3–19. Springer, 2024. doi:10.1007/978-981-97-5562-2_1.

[227] Jiajun Zhou, Chenkai Hu, Jianlei Chi, Jiajing Wu, Meng Shen, and Qi Xuan. Behavior-aware account deanonymization on Ethereum interaction graph. *IEEE Transactions on Information Forensics and Security*, 17:3433–3448, 2022. doi:10.1109/TIFS.2022.3208471.

[228] Patrick Züst, Tejaswi Nadahalli, and Roger Wattenhofer. Analyzing and preventing sandwich attacks in Ethereum. Bachelor's thesis, ETH Zürich. https:

//pub.tik.ee.ethz.ch/students/2021-FS/BA-2021-07.pdf, 2021.

A Extending our Framework beyond RPCs

Sections 4.2–4.4 instantiate the attack framework for the non-colluding RPC setting. The same abstraction extends to adversarial intermediaries elsewhere in the blockchain stack. This section lifts the RPC-specific notation to a stack-wide leakage model and shows how prior siloed results–wallet leakage [79], transaction-graph deanonymization, relay censorship, mempool sandwich attacks, and sequencer MEV–fit into one language.

Layers of the blockchain stack. We model blockchain systems as a layered stack [26, 178] as shown in Fig. 6, the layers being a) *L0*: peer-to-peer networking, node discovery, and message propagation [66, 93, 140], b) *L1*: base-chain consensus and computation: agreement on an ordered ledger and execution of the state-transition function [153, 220], (c) *L2*: protocols built above L1 to improve throughput, latency, or cost while relying on L1 for settlement or security [97, 154], (d) *Access layer*: wallets, APIs, scripts, SDKs, and dApps through which users access blockchain functionality [130, 139]. Validators are L1-native, builders, and relays are L1-facing [35, 99]; sequencers and Bitcoin Lightning nodes are L2-native [167]; wallets are access-layer-native; and RPC services sit at the boundary between the access layer (RPC API) and the underlying L1 or L2 chain (RPC node) [130, 139].

Intermediaries. Let Q be the set of blockchain-stack intermediaries: wallets, RPC providers, P2P nodes, searchers, builders, relays, validators, sequencers, and L2 infrastructure. Fix $q \in Q$ as the primary adversary. The RPC-specific framework in Section 3 studies the special case $q = \mathsf{rpc}$; the general model lets q be any intermediary that receives, forwards, transforms, orders, or exposes user-originating blockchain interactions.

Exchange and Exchange Log. Let F range over client-requested blockchain operations beyond the RPC interface: state read, transaction submission, transaction propagation, block construction, block proposal, sequencing. An *exchange* is the functionality-specific message flow r_F that q participates in or mediates i.e. the messages q receives from and sends to adjacent parties in order to realize F. An *exchange log* $R = (r_{F_1}, \ldots, r_{F_m})$ is a time-ordered sequence of exchanges spanning functionalities and stack layers.

Adversarial View. Each exchange r_F induces a party-specific adversarial view $L_q(r_F)$: the components of the exchange that q actually observes. The stack-wide view extends the RPC tuple from §3 with layer-specific fields:

$$L_q(r_F) = \big(\ell_{\mathsf{time}}, \ell_{\mathsf{vol}}, \ell_{\mathsf{net}}, \ell_{\mathsf{app}}, \ell_{\mathsf{L0}}, \ell_{\mathsf{L1}}, \ell_{\mathsf{L2}}, \ell_{\mathsf{public}}\big).$$

Here ℓ_{L0} is P2P-layer information such as peer connectivity, propagation metadata, and gossip timing; ℓ_{L1} is base-chain ordering, block construction, and execution data; ℓ_{L2} is L2 sequencing, batching, and execution data; and ℓ_{public} is public on-chain data. ℓ_{time}, ℓ_{vol}, ℓ_{net}, and ℓ_{app} retain their RPC-specific meanings. As before, each ℓ_c represents the leakage from the underlying full exchange field.

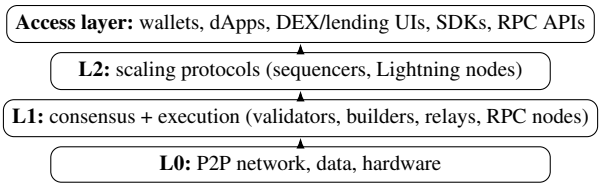


Figure 6: Layers of the blockchain stack. RPC services sit at the boundary between the access layer and the underlying L1/L2 chain.

Leakage profile. For an exchange log $R = (r_{F_1}, \ldots, r_{F_m})$, the party-specific leakage profile is $L_q(R) = \big(L_q(r_{F_1}), \ldots, L_q(r_{F_m})\big)$. This profile captures cross-interaction correlations: repeated wallet telemetry, P2P propagation fingerprints, builder-side order-flow patterns, relay acceptance behavior, validator proposal behavior, and L2 sequencing choices.

Collusion is common in the access and block-production stack: a wallet may share infrastructure with an RPC provider, an RPC may route to preferred builders, a builder may use particular relays, and an L2 RPC may be economically or operationally tied to a sequencer. To model this, we characterize leakage under collusion.

Collusion-composed leakage profile. Let $Q' \subseteq Q \setminus \{q\}$ be the colluding set. The composed leakage view is $L^{\mathsf{coll}}_{q,Q'}(R) = \big(L_q(R), \{L_s(R) : s \in Q'\}\big)$. The non-colluding RPC model used in the attack instantiations is the special case $L^{\mathsf{coll}}_{\mathsf{rpc},\emptyset}(R) = L_{\mathsf{rpc}}(R)$.

Adversary Model. The adversary model generalizes from $M = (Q', \mathsf{Cap})$ to $M = (q, Q', \mathsf{Cap})$, where q is the primary adversarial intermediary, Q' is the colluding set, and Cap is a set of active capabilities: $\mathsf{Cap} \subseteq \{C_{\mathsf{drop}}, C_{\mathsf{delay}}, C_{\mathsf{inject}}, C_{\mathsf{poison}}, C_{\mathsf{route}}\}$. An attack remains $\Pi = (M, K, A, G)$, with A now taking $(L^{\mathsf{coll}}_{q,Q'}(R), \mathsf{Cap}, K)$ as input. This is the same template as Section 3; only the adversarial viewpoint has changed from an RPC-only view to a stack-wide collusion-composed view.

Mapping Prior Attacks into the Framework. Table 3 maps representative prior attacks into the generalized tuple. The purpose is to show that the same leakage view and attack tuple are sufficient to describe attacks that are usually studied separately. Wallet telemetry leakage can be studied as an instance in which the primary adversary is the wallet that learns ℓ^{app} and ℓ^{net} that includes device/session metadata and destination RPC. In transaction graph deanonymization, any party can be the primary adversary, as the attack stems from ℓ_{public}. Relay censorship and builder/searcher sandwiching take into account leakage in the L1 block-production pipeline. Sequencer

censorship and reordering take into account leakage in the L2 layer.

B Customer-Specific Leakage Impact

We instantiate the leakage components of Section 3 for three representative RPC customer segments, identifying the customer-sensitive inferences that follow (Table 4). To our knowledge, this is the first systematic treatment of RPC leakage from the customer's perspective: prior work documents RPC-side leakage [79, 130, 139, 214] and separately documents customer-specific privacy risk empirically [208, 215, 222], but treats the two in isolation. As Table 4 shows, the same addr_r can support different sensitive inferences when attributed to different customers. These inferences may additionally depend on request methods, timing, transaction parameters, and auxiliary on-chain state or public labels, as specified in the table.

Trading firms and MEV searchers. MEV-discovery systems repeatedly query decentralized-exchange state to identify opportunities [45], exposing to the RPC the firm's *strategy input set*: the pools, assets, accounts, and other variables it monitors for opportunities [45]. Candidate transaction simulations disclose the contemplated routes, amounts, and slippage bounds under evaluation, including candidates that never appear on-chain [45]. Variants tested before submission reveal the search trajectory and parameter neighbourhood around the submitted candidate, approximating the conditions under which the strategy actually executes [45]. Finally, the submitted transaction along with the resulting on-chain state lets an observer classify the executed MEV strategy and reconstruct its resulting asset flows [138, 171]. Taken together, we infer that these leakage components could let an RPC reconstruct a firm's proprietary trading logic.

Centralized exchanges. Address-specific balance and history queries attributed to an exchange reveal its RPC-observed watchlist [168]. Where the exchange assigns a separate deposit address per customer, a pattern heuristically identifiable on Ethereum [208], growth in the number of newly observed deposit addresses provides a proxy for customer growth, if not an exact customer count [133]; exposing what is normally confidential competitive intelligence about a rival's user base. Combining query and transaction patterns with on-chain history can cluster exchange-controlled addresses, distinguish deposit, hot, and cold-wallet roles, and identify deposit or withdrawal flows, following the same address-tagging and role-classification methodology demonstrated for exchange wallets on Bitcoin [133, 223]. Their balances yield an observed asset inventory limited to the addresses known to the RPC [133]. Where the target contract is known, a simulated large transfer or contract call exposes a prospective swap, bridge transfer, liquidity movement, or other operation before submission. Recurring sweeps—periodic consolidation of deposit-address balances into a central hot or cold wallet—and address-use sequences such as hot-to-cold transfers reveal the timing and pattern of exchange wallet-management activity, again mirroring the operational fingerprints observed for exchange wallets in practice [133, 223].

Data, analytics, and compliance firms. When requests are attributable to an analytics customer, repeatedly queried addresses and transactions expose its current graph-search targets, an access-pattern leakage well documented outside blockchain settings [111] and realized concretely by blockchain graph-tracing tools [222]. A sudden increase in query frequency exposes increased analyst attention to a target or cluster. The order in which the analyst queries transfers, for example, following A to B to C, reveals the path being explored through the transaction graph [222]. Repeated exploration of a graph containing public entity or risk labels can further suggest an objective such as illicit-fund tracing or asset recovery [222], or anti-money-laundering (AML) screening [218], although the objective remains inferred rather than directly observed, effectively exposing an active investigation to the infrastructure carrying it.

We selected these three customer segments to span the range of relationships an RPC provider has with its customers. Extending this to further customer classes, such as cross-chain bridges and autonomous on-chain agents, is discussed as future work in Section 5.

C MEV Countermeasures

Table 5 classifies the deployed and proposed sandwich-MEV countermeasures discussed in Section 4.4 along the leakage class (Type) and execution setting.

D Evaluation

Sections 3 and 4 analyze how each Type affects each attack; we now quantify that empirically. To our knowledge, we build the first dataset of client–RPC interactions realistic enough to meaningfully evaluate leakage-based attacks. We follow a ledger-grounded methodology [214], anchoring each real on-chain transaction to the write step of a user's wallet workflow, then pass state read requests from the same workflow through the RPC gateway, backed by Ganache [202], which simulates a blockchain locally. We run deanonymization, censorship, and sandwich MEV algorithms, measuring attack success for each Type. All experiments run on a Chameleon Cloud [118] bare-metal `compute_skylake` node (dual Intel Xeon Gold 6126 CPUs, 24 cores/48 threads, 192 GiB RAM) at CHI@UC, running Ubuntu 18.04.

Deanonymization Setup. In addition to wallet workflows anchored in real on-chain transactions, we also construct portfolio refresh (defined in Section 2) wallet workflows and replay them against a real RPC gateway to produce a real interaction log, shared by the deanonymization and censorship

Table 3: Representative stack-wide attacks as instantiations of the generalized leakage-and-attack tuple.

Attack	**Primary** q	Cap	**Key adversarial view**	**Auxiliary** K	**Goal** G
Wallet telemetry leakage [79]	wallet	$\emptyset$	$\ell_{\mathsf{app}}, \ell_{\mathsf{net}} = \{$device/session metadata, destination RPC$\}$	wallet UI/session context	Link address activity to a wallet session or device profile
Access-layer timing deanonymization [214]	proxy / router	$\emptyset$	$\ell_{\mathsf{net}}, \ell_{\mathsf{time}}, \ell_{\mathsf{vol}}, \ell_{\mathsf{public}}$	wallet heuristics	IP-to-address linkage
Transaction-graph deanonymization [9, 145, 208]	any party	$\emptyset$	ℓ_{public}	common-input, change-address, and behavioral heuristics [22]	Cluster addresses to one entity
Relay or builder censorship [205, 209]	relay / builder	C_{drop}	$\ell_{\mathsf{L1}} = \{$addr, contract$\}$	sanctioned-address or contract list	Selectively drop transactions
Sandwich attack [61, 171, 228]	searcher / builder	C_{inject}	$\ell_{\mathsf{L1}} = \{$pending swap parameters, slippage bound$\}$	pool reserves and fee tier	Force victim swap to execute at a worse AMM price
Sequencer censorship [41]	L2 sequencer	C_{drop}	$\ell_{\mathsf{L2}} = \{$addr, contract$\}$	exclusion list or policy predicate	Exclude specific users or transactions
Sequencer MEV reordering [14, 74, 201]	L2 sequencer	$C_{\mathsf{inject}}, C_{\mathsf{drop}}, C_{\mathsf{delay}}$	$\ell_{\mathsf{L2}} = \{$tx parameters$\}$	$\perp$	Front-run or sandwich without a public auction

evaluations and taking 48 hours of wall-clock time to replay; full parameters are in Table 9. Two types of users, *casual* users and *traders* (550 in total), differ in how many wallet addresses and source IPs each controls, with a fraction of IPs shared across users to model shared households. We assign each address a deterministic balance. Balances are spaced deterministically so that every user's total is distinct, making the Type 2–4 wealth prior exact and uniquely identifying; our results therefore establish feasibility under such a prior rather than under noisy or colliding wealth estimates. The Type 1 prior is a synthetic user-to-IP-set mapping. All users connect through an ethers.js-based wallet client that reproduces MetaMask's documented RPC timing (block polling, balance-read bursts; Table 9) rather than the MetaMask browser extension itself, and this timing shapes the Type 2 attack. At Type 2 a rotating pool of Tor/VPN exit IPs replaces each user's stable IP set (both the sharing and the Tor substitution are applied as synthetic `x-source-ip` labels layered onto the trace, not real network-layer changes to the real-RPC requests), due to the infeasibility of setting up 550 different machines with different IPs. The Type 0–5 leakage profiles are then derived from this ground-truth log by suppressing the components specified by $L_{\mathsf{rpc}}(R)$.

Deanonymization Results. Table 6 reports accuracy, precision, and recall at each Type. Types 0–4 achieve accuracy, precision and recall of at least 99%, while the same metrics are negligible for Type 5. At Type 1, the drop in accuracy from 100% to 99.63% is due to the 15% of IPs shared across users to model shared households: when two users' IP signatures overlap, the device/TLS/session-fingerprint refinement does not always separate their clusters, so a small fraction of addresses are attributed to the wrong user. Figure 7 traces accuracy and precision/recall as the burst gap threshold Δ varies. For $\Delta \lesssim 12$ ms, each portfolio refresh is fragmented across

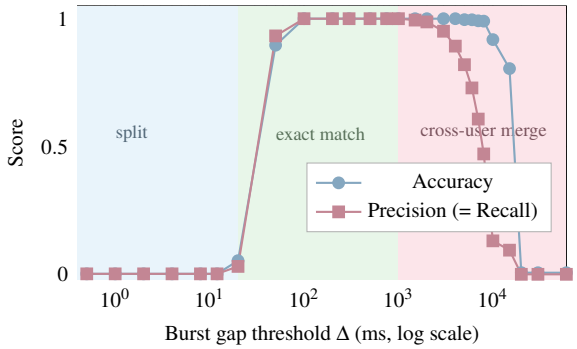


Figure 7: Type 2–4 burst-clustering deanonymization accuracy and precision/recall vs gap threshold Δ

multiple bursts, which explains the negligible accuracy, precision, and recall (0.18%). For $100\,\text{ms} \leq \Delta \leq 1\,\text{s}$, each user's portfolio refresh fits in a single burst and the summed balance matches the user's total exactly, so accuracy, precision, and recall are all 100%; accuracy holds at 100% through $\Delta = 4\,\text{s}$ while precision and recall begin to decline (89% at 4 s). For $\Delta \geq 5\,\text{s}$, consecutive users' portfolio refreshes start to merge into one burst: precision and recall decline first (82% at 5 s, 13% at 10 s) while accuracy stays high longer (99.6% at 5 s, 91.8% at 10 s), and by $\Delta = 20\,\text{s}$ accuracy has fallen to 0.65% and precision/recall to 0%.

Censorship Setup. Censorship runs on the same trace, adding synthetic sanctioned sets sized to plausible real-world proportions (Table 9): a sanctioned-user list $\mathcal{S}_{\mathsf{SDN}}$, blocked jurisdictions $\mathcal{J}_{\mathsf{OFAC}}$ by IP range, and the deployed mock Uniswap V2 router as the sole sanctioned contract $\mathcal{S}_{\mathsf{SC}}$ (modeling the Tornado Cash precedent). At Types 2–4 the censor evaluates the jurisdiction predicate against the Tor exit IP rather than

Table 4: Customer-specific RPC leakage: components of $L_{\mathsf{rpc}}(R)$ (in parentheses) and the resulting sensitive inferences.

Customer	Leakage to RPC	Derived inference
Trading firms / MEV searchers	Repeated reads of particular pools, tokens, contracts, or accounts, including balances, positions, reserves, prices, and contract state ($\ell_{\mathsf{id}} + \mathsf{addr}_{\mathsf{r}} + F$)	**Strategy input set**: MEV-discovery systems explicitly search DEX state and mutate strategy inputs [45].
	Candidate swaps simulated with particular routes, amounts, and slippage bounds ($\ell_{\mathsf{id}} + \mathsf{tx}_{\mathsf{r}}$)	**Candidate strategy actions**: Routes and parameter choices evaluated before submission, including candidates never written on-chain [45].
	Similar candidates with varied parameters are simulated immediately before a transaction is submitted ($\mathsf{tx}_{\mathsf{r}} + \mathsf{tx}_{\mathsf{w}} + \ell_{\mathsf{time}}$)	**Search trajectory / execution boundary**: The tested parameter sequence and the neighbourhood of the submitted candidate, which approximate the conditions under which the strategy executes [45].
	Submitted transactions linked to firm-controlled addresses ($\ell_{\mathsf{id}} + \mathsf{addr}_{\mathsf{w}} + \mathsf{tx}_{\mathsf{w}}$+ on-chain state)	**Executed strategy and position changes**: On-chain transactions permit classification of executed MEV strategies and reconstruction of the resulting asset flows [138, 171].
Centralized exchanges	Same customer repeatedly queries balances or transaction histories for many addresses ($\ell_{\mathsf{id}} + \mathsf{addr}_{\mathsf{r}}$+ on-chain state)	**Exchange watchlist and competitive intelligence**: Queried addresses form the RPC-observed watchlist [168]. If the exchange assigns each customer a separate deposit address [208], growth in their number provides a proxy for exchange-user growth, hence competitive intelligence [133].
	Repeated query and transaction patterns involving exchange-controlled addresses ($\mathsf{addr}_{\mathsf{r}} + \mathsf{addr}_{\mathsf{w}} + \mathsf{tx}_{\mathsf{w}} + \ell_{\mathsf{time}}$ + on-chain state)	**Grouping and wallet role**: Transaction counts, balances, timing, and transfer patterns can support clustering exchange addresses, classifying them as deposit, hot, or cold wallets, and distinguishing deposits from withdrawals. [133, 223].
	Exchange-associated addresses and their balances ($\mathsf{addr}_{\mathsf{r}}$+ on-chain state)	**Observed asset inventory**: Balances across tagged exchange addresses estimate holdings attributable to those addresses [133].
	A large transfer or contract call is simulated ($\mathsf{tx}_{\mathsf{r}} + F$)	**Prospective action**: The target, value, function selector, and decoded contract arguments expose the simulated transfer, swap, bridge, or liquidity-management call and its parameters.
	Recurring sweeps and wallet-management cycles ($\ell_{\mathsf{time}} + \mathsf{addr}_{\mathsf{r}}/\mathsf{addr}_{\mathsf{w}}$)	**Operational cycle**: Transfer timing and patterns reveal when exchange consolidates deposits or moves funds among its wallets. [133, 223].
Data / analytics / compliance firms	A known analytics customer repeatedly queries particular addresses or transactions ($\ell_{\mathsf{id}} + \mathsf{addr}_{\mathsf{r}}$)	**Investigation target set**: Repeatedly selected addresses and transactions expose the targets of the analyst's graph search [111, 222].
	Querying of an address or address cluster suddenly intensifies ($\ell_{\mathsf{id}} + \mathsf{addr}_{\mathsf{r}} + \ell_{\mathsf{time}}$)	**Investigation escalation**: A higher query rate exposes increased analyst attention to the address or cluster.
	An investigator follows transfers from address A to B to C ($\ell_{\mathsf{id}} + \mathsf{addr}_{\mathsf{r}} + \ell_{\mathsf{time}}$ + on-chain state)	**Investigation path**: The ordered queries reconstruct the path followed through the on-chain transfer graph [222].
	A particular labelled transaction graph is repeatedly explored ($\ell_{\mathsf{id}} + \mathsf{addr}_{\mathsf{r}}$+ on-chain state and public labels)	**Likely analytical objective**: Illicit-fund tracing or asset recovery [222], or AML screening [218], depending on the public labels and paths explored.

Table 5: MEV countermeasures in the RPC leakage and execution framework.

Countermeasure class	Deployment status	Leakage class	Execution setting
MEV auction platforms/ private routing	Most widely adopted: MEV-Boost [84], Flashbots Protect [83], MEV Blocker [58], and bloXroute Protect [24]	Type 0–Type 3: hides from public mempool, not from the receiving RPC.	E_0 (RPC can submit bundles to builder)
Time-priority ordering	Not on Ethereum L1: Arbitrum Timeboost [13]	Type 0–Type 3: no RPC leakage reduction.	E_0 if all users use RPC, E_1 else (RPC can still manipulate transaction submission time via C_{delay} or C_{route})
Fair ordering	Proved to not mitigate MEV for arbitrary payoff functions [46]	Type 0–Type 3: no RPC leakage reduction.	E_2
Encrypted mempools	research / not deployed: Aptos TrX [78] (governance approval pending)	Type 4 (only if wallet-side encryption)	E_0 (if encrypted transaction identified correctly based on simulation)
Encrypted mempools+ TEE/ORAM	research	Type 6 (only if wallet-side encryption)	N/A (profit predicate uncomputable)

the user's real IP. By ground truth, 37.55% of the 16,875 transaction-submissions match at least one predicate.

Table 6: Deanonymization results along leakage classes.

Architecture type	Accuracy %	Precision %	Recall %
Type 0	100.00	100.00	100.00
Type 1	99.63	99.23	99.45
Types 2–4	100.00	100.00	100.00
Type 5[a]	0.12	0.06	0.18

[a] In Type 5, every address is assigned to a random user.

Censorship Results. Table 7 reports selectivity, accuracy, and false-positive rate per Type. At Type 0, the targeting is perfect. At Type 1, IP-set deanonymization introduces a small number of user entity misattributions (3 false positives). From Type 2, jurisdiction predicate evaluation breaks, due to the predicate applied on the Tor exit IP instead of the client IP, which drops accuracy to 78.79%, and contributes to a false-positive rate of 4.81%. In Type 3, the contract predicate is computed perfectly, and the user entity is derived via the preceding portfolio-refresh burst. At Type 4, the contract predicate evaluation likewise is derived via the preceding `eth_estimateGas` within a gap of $\sim$1.5 s. At Type 5, addr_r is suppressed, so user entity can no longer be recovered from the portfolio-refresh burst; only contract predicate evaluation survives, still computed from the visible tx_r of the preceding simulation, so contract-level censorship persists. Only at Type 6, once tx_r is hidden, does contract predicate evaluation break as well.

Table 7: Censorship results across leakage classes

Architecture type	Selectivity %	Accuracy %	FPR %
Type 0	37.55	100.00	0.00
Type 1	37.58	100.00	0.06
Type 2-4	32.59	78.79	4.81
Type 5	14.09	29.51	4.81

Sandwich MEV Setup. The MEV profile replays 56,348 recent Uniswap-v2 swaps (details in Table 9). The trace records trade amounts but not pool reserves, so we synthesize each pool's depth from its first trade: at the first observed swap we set the input-side reserve to $D \cdot \texttt{amount\_in}_1$ with $D = 200$, making that swap 0.5% of the reserve; a price impact representative of a liquid pool. Subsequent swaps update reserves deterministically. Slippage bounds range from 0.01% to 2% (drawn synthetically). Targets are selected in two stages: we first sample 80% of the swaps as RPC-observed targeted swaps; the remaining 20% form the *background pool* that models unrelated transactions interleaving between attacker submissions under E_1. We then classify each targeted swap as sandwichable or unsandwichable under the simulated reserve, slippage, and fee model. For each sandwichable candidate the simulator computes the front-run size by binary search to maximize attacker profit subject to $\Delta^{\min}_{\text{out}}$. Under E_1, each

Table 8: Sandwich-attack results across execution settings.

Setting	Positive-profit candidates	Non-positive candidates	Profit rate	Total profit$_a$ (USD)
E_0	19,713	3,361	85.4%	\$6.57M
E_1 (λ=1)	17,715	5,359	76.8%	\$7.76M
E_2	2,976	20,098	12.9%	−\$1.85M

[a] We convert simulated token profits to USD using fixed market-price snapshots for interpretability; these values quantify extractable value under our synthetic pool assumptions, not historical extraction opportunities.

inserted background swap uses input size 0.1% of the reserve with Poisson arrivals at intensity λ; the table reports λ=1, and Figure 8 traces $\lambda \in \{0, 0.5, 1, 2, 5, 10, 20\}$. USD totals price the major tokens (WETH, WBTC, and the stablecoins) directly from CoinGecko's daily close at the 2026-03-14 trace midpoint. A long-tail token is not listed, so it is valued through its counter-token — the major token paired with it in the same swap pool. We convert the long-tail amount at the pool's spot rate (the reserve ratio `reserve_out`/`reserve_in` at that swap, i.e. how many counter-tokens one input token is worth) and multiply by the counter-token's snapshot price. Since each of E_0, E_1, and E_2 takes nearly 45 hours of wall-clock time over the full trace ($\approx$ 135 hours across the three settings), we simulate the pool rather than deploying every swap and replaying it live through the RPC gateway; the reserve, slippage, and background-swap models are therefore synthetic, while the swap sequence itself is real.

Sandwich MEV Results. In E_0, 85.4% of sandwichable targets yield profit, totaling \$6.57M; the rest fall to tight slippage bounds or unfavorable reserve ratios. At E_1 (λ=1), the profit rate drops to 76.8% but aggregate profit *rises* to \$7.76M; same-direction interleaving substitutes for reverted intended victims, amplifying tx_B's capture. Figure 8 traces the λ sweep: profit rate decays monotonically to 62.9% by λ=20, while total profit stays in the \$6.6–9.9M range. Under E_2, half of all permutations execute the back-run before the front-run; the attacker pays the front-run amount, yielding negative measured profit. These $\sim$11,500 forfeited attempts overwhelm the few profitable cases, leaving net aggregate profit at −\$1.85M. Table 8 reports outcomes over the 23,074 sandwichable targeted swaps, not all 44,986 RPC-observed targeted swaps.

E Formal Attack Descriptions and Pseudocode

We first give more formal descriptions of each attack instantiated in Section 4, which set up the notation, then the full pseudocode of each algorithm A: deanonymization at Types 0, 1, and 2–4 (Alg. 1–3), censorship (Alg. 4), and sandwich MEV (Alg. 5).

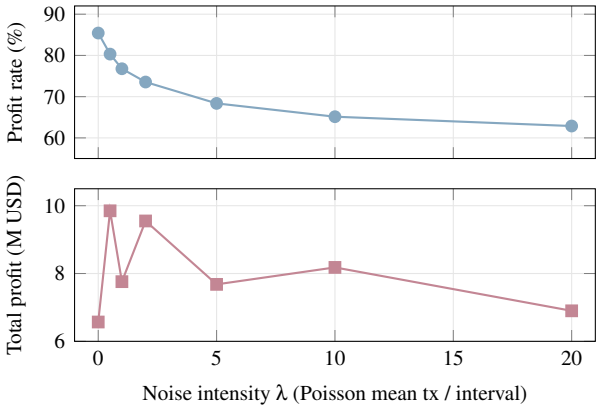


Figure 8: E_1 sandwich attack with varying interleaving-noise intensity λ.

Algorithm 1 Deanonymization at Type 0 (API-key clustering)

api_keys $\leftarrow \{k_i \mid 1 \le i \le n\}$
for each $k \in$ api_keys **do**
  $G(k) \leftarrow \{r_i \mid k_i = k\}$
  $\hat{S}(k) \leftarrow \bigcup_{r \in G_k} \mathsf{ExtractAddress}(\ell_{\mathsf{app}}(r))$
  $C(k) \leftarrow \emptyset$
  for each $addr \in \hat{S}(k)$ **do**
    $bal \leftarrow \mathsf{GetBalance}(addr)$
    $C(k) \leftarrow C(k) \cup \{(addr, bal)\}$
  end for
  $\hat{u}(k) \leftarrow \mathsf{ObtainUser}(k)$
end for
Output: $\{(\hat{u}(k), C(k))\}_{k \in \text{api\_keys}}$

E.1 Deanonymization

Let $\mathcal{U}$ be the universe of real-world users and $S(u)$ the true set of addresses controlled by $u \in \mathcal{U}$, so $S = \bigcup_{u \in \mathcal{U}} S(u)$. Let $\mathsf{Bal}(a)$ be the public on-chain balance of a. The goal G partitions observed addresses into clusters $C_1, \ldots, C_m$, assigning each C_i to a user $\hat{u}_i$ (the address-to-user mapping $\hat{u}$) and recovering the portfolio $\{(a, \mathsf{Bal}(a)) : a \in C_i\}$.

Algorithm 1. Prior knowledge $K = \perp$.

1. cluster interactions by API key or account identifier
2. in each cluster, extract every address appearing in read queries, transaction envelopes and parameters
3. query the public ledger for each address's balance
4. resolve each cluster to a user via the provider's account record (ObtainUser)

Algorithm 2. Prior knowledge $K = \{(u, \mathsf{IP}(u)) : u \in \mathcal{U}\}$, where $\mathsf{IP}(u)$ is the IP set associated with u

1. extract address–IP edges from $L_{\mathsf{rpc}}(R)$
2. build $\mathsf{ips}[a]$, the set of IPs observed with address a
3. cluster addresses with the (near-)same IP signature

Table 9: Configuration profiles for the deanonymization/censorship and MEV evaluations.

Deanonymization / Censorship profile	
Trace window	14.9M Ethereum public transactions from AWS dated 2026-03-11 – 2026-03-17; real-RPC replay for deanonymization and censorship combined ran 48 hours wall-clock
Interaction log	Total 1,536,461 (176,897 `eth_getBalance`, 16,877 `eth_estimateGas`, 16,875 `eth_sendRawTransaction`)
Users	500 casual + 50 traders (550 total)
Addresses / user	2–5 (casual), 10–15 (traders)
IPs / user	1–2 (casual), 2–4 (traders)
IP% shared between users	15% (synthetic `x-source-ip` label)
Tor exit pool	100 IPs (synthetic label substitution)
Wallet client	ethers.js client emulating MetaMask's RPC timing (intra-burst gap 0.7–1.8 ms; inter-step gap 1.2–2.5 s), derived from a Ganache HD-wallet mnemonic
User Balances	Casual $\in [0.05, 30]$ ETH, Trader $\in [80, 2000]$ ETH; unique by construction
Δ	default 100 ms; fine sweep 0.5–12 ms, coarse sweep to 60 s (Fig. 7)
Sanctioned sets	$\lvert\mathcal{S}_{\mathsf{SDN}}\rvert = 110$ (20% of users), $\lvert\mathcal{I}_{\mathsf{OFAC}}\rvert = 137$ of 984 distinct user IPs (13.9%, target 15%), $\mathcal{S}_{\mathsf{SC}}$ = deployed Uniswap mock router; 6,336 of 16,875 writes (37.55%) match at least one predicate
MEV profile	
Trace window	56,348 Uniswap public transactions (AWS) 2026-03-11 – 2026-03-17 (7 days)
Targeted swaps (80%)	44,986 (sandwichable: 23,074)
Background swaps (20%)	11,362 swaps
Reserve init	$D = 200$; the input-side reserve at the first observed swap of each pool is set to $D \cdot$ `amount_in`$_1$, then updated deterministically from later swaps
Fee model	fee rate $\varphi = 0.003$ (0.3%)
Slippage range	$\mathcal{U}(0.01\%, 2.00\%)$ (drawn synthetically; not recoverable from Swap/Transfer logs alone)
Interleaving (E_1)	Poisson $\lambda = 1.0$ for Table 8; sweep $\lambda \in \{0, 0.5, 1, 2, 5, 10, 20\}$ for Fig. 8; background tx size = 0.1% of reserve
Runtime	E_0, E_1, E_2 each ≈ 45 hours wall-clock in simulation (≈ 135 hours total)

4. refine (Refine) by dropping transient IPs, splitting on device/TLS/session fingerprints, and merging identical address sets
5. assign cluster c to the user $\hat{u}$ whose IP footprint is a superset of c

Algorithm 3. Prior knowledge $K = \{(u, w(u)) : u \in \mathcal{U}\}$, with ranked wealth $w(u)$:

1. extract timestamps and queried addresses for all `eth_getBalance` reads
2. sort by timestamp and partition into bursts using a gap threshold Δ
3. let each burst induce a candidate cluster and attach balances via public-ledger reads
4. refine (Refine) by intersecting address sets across bursts of the same user and merge identical clusters

Algorithm 2 Deanonymization at Type 1 (IP-set clustering)

$E \leftarrow \{(addr, ip) \mid ip = \ell_{\mathsf{net}}(r_i).\mathsf{ip\_address},\ addr \in \mathsf{ExtractAddress}(\ell_{\mathsf{app}}(r_i))\}$
$\mathsf{ips} \leftarrow \{\}$
for each $(addr, ip) \in E$ **do**
  $\mathsf{ips}[addr] \leftarrow \mathsf{ips}[addr] \cup \{ip\}$
end for
$C \leftarrow \{\}$
for each address $addr \in \mathsf{ips}$ **do**
  $c \leftarrow \mathsf{ips}[addr]$
  $bal \leftarrow \mathsf{GetBalance}(addr)$
  $C[c] \leftarrow C[c] \cup \{addr, bal\}$
end for
$C \leftarrow \mathsf{Refine}(C)$
for each key $c \in C$ **do**
  Sample $\hat{u}(c)$ from $\mathcal{U}$ s.t. $c \subseteq \mathsf{IP}(\hat{u}(c))$
end for
Output: $\{(\hat{u}(c), C[c])\}_{c \in C}$

(MergeIdentical), since repeated portfolio scans by one persona produce duplicate bursts

5. align clusters to K by total balance (Tot_Bal) in two passes: exact total-balance matches first, then descending-rank matches for the remainder

E.2 Censorship

Let R_{pre} be the interaction stream observed before censorship and $T \subseteq R_{\mathsf{pre}}$ the target set the adversary intends to suppress. Prior knowledge $K = (S_{\mathsf{SDN}}, J_{\mathsf{OFAC}}, S_{\mathsf{SC}}, f)$: a sanctioned-user list S_{SDN}, blocked jurisdictions J_{OFAC} (by IP range), blocked contracts S_{SC}, and an address-to-user map f from deanonymization. Three predicates index the leakage they consume: $\psi_{\mathsf{entity}}(r)$ fires when a_{from}, deanonymized via f, lies in S_{SDN}; $\psi_{\mathsf{juris}}(r)$ when $\mathsf{srcIP}(r)$ falls in J_{OFAC}; $\psi_{\mathsf{contract}}(r)$ when $a_{\mathsf{to}} \in S_{\mathsf{SC}}$ or some contract in the preceding read burst $\mathcal{B}(r)$ (an `eth_call`/`eth_estimateGas` simulation) lies in S_{SC}. The target set is $T = \{r : \psi_{\mathsf{entity}}(r) \vee \psi_{\mathsf{juris}}(r) \vee \psi_{\mathsf{contract}}(r)\}$, and G drops every $r \in T$ while forwarding the rest.

Algorithm 4 For each write $r_{\textsc{SendTx}} \in R_{\mathsf{pre}}$:

1. extract a_{from}, $\mathsf{srcIP}(r)$, and a_{to} from $L_{\mathsf{rpc}}(r)$
2. resolve the sender to a user via f
3. evaluate $\psi_{\mathsf{entity}}, \psi_{\mathsf{juris}}, \psi_{\mathsf{contract}}$ against $(S_{\mathsf{SDN}}, J_{\mathsf{OFAC}}, S_{\mathsf{SC}})$
4. if any is true, invoke C_{drop} to suppress r, otherwise forward r unchanged

E.3 Sandwich MEV

A victim swaps token X for token Y on a constant-product pool with reserves (r_X, r_Y) and fee rate φ, input Δ_{in}, and minimum acceptable output $\Delta_{\mathsf{out}}^{\min}$ (derived from slippage tolerance). Prior knowledge $K = (\varphi, r_X, r_Y)$, both publicly readable. We write $\mathsf{Out}(r_X, r_Y, \Delta, \varphi)$ for the resulting constant-product swap output. The profit predicate $\psi_{\mathsf{profit}}(r)$ fires when the maximum realizable sandwich profit $\Pi^*(r)$ is strictly positive, computable from the swap parameters in $\mathsf{tx_w}$ or, when those are hidden, from $\mathsf{tx_r}$ in a preceding `eth_call`/`eth_estimateGas` simulation. The goal G is to realize sandwich profit.

Algorithm 5 For each visible swap r with $\psi_{\mathsf{profit}}(r) = 1$:

1. read the pool's current reserves (r_X, r_Y)
2. binary-search the largest front-run input f^* that keeps the victim's output above $\Delta_{\mathsf{out}}^{\min}$
3. compute the back-run output and net profit, accounting for gas, the priority-fee premium, and any builder or relay payment needed to secure ordering
4. if net profit is positive, construct tx_F and tx_B, control victim propagation via $C_{\mathsf{delay}}/C_{\mathsf{drop}}$, bundle (tx_F, tx_V, tx_B) via C_{inject}, and route the bundle to a builder via C_{route}.

Algorithm 3 Deanonymization at Type 2-4 (burst clustering)

Initialize list $\mathsf{Q} \leftarrow [\,]$
for each `eth_getBalance` interaction r_i **do**
  $t \leftarrow \mathsf{ExtractTime}(\ell_{\mathsf{time}}(r_i))$;
  $addr \leftarrow \mathsf{ExtractAddress}(\ell_{\mathsf{app}}(r_i))$
  Append $(t, addr)$ to Q
end for
Sort Q by timestamp t
$\mathcal{B} \leftarrow \mathsf{PartitionIntoBursts}(\mathsf{Q}, \Delta)$
$C = \emptyset$
for each burst $B \in \mathcal{B}$ **do**
  for each $(t, addr) \in B$ **do**
    $bal \leftarrow \mathsf{GetBalance}(addr)$
    $C[B] \leftarrow C[B] \cup \{(addr, bal)\}$
  end for
end for
$C \leftarrow \mathsf{Refine}(C)$; $C \leftarrow \mathsf{MergeIdentical}(C)$
$\mathsf{Tot\_Bal}(c) := \sum_{(addr,bal) \in C[c]} bal$ for each cluster key c
$\mathsf{used} \leftarrow \emptyset$, $\mathsf{Unmatched} \leftarrow [\,]$
for each cluster key $c \in C$ **do**
  if $\exists u \in \mathcal{U} \setminus \mathsf{used}$ with $w(u) = \mathsf{Tot\_Bal}(c)$ **then**
    $\hat{u}(c) \leftarrow u$; $\mathsf{used} \leftarrow \mathsf{used} \cup \{u\}$
  else
    append c to $\mathsf{Unmatched}$
  end if
end for
Sort $\mathsf{Unmatched}$ by $\mathsf{Tot\_Bal}(\cdot)$ descending; let $u'_1, \ldots, u'_{|\mathsf{Unmatched}|}$ be the still-unassigned users in $\mathcal{U} \setminus \mathsf{used}$ in descending order of $w(\cdot)$
for $j \leftarrow 1$ **to** $|\mathsf{Unmatched}|$ **do**
  $\hat{u}(\mathsf{Unmatched}[j]) \leftarrow u'_j$
end for
Output: $\{(\hat{u}(c), C[c])\}_{c \in C}$

Algorithm 4 Censorship

for each `eth_sendRawTransaction` r_i **do**
  $a_{\mathsf{from}} \leftarrow \mathsf{ExtractFromAddress}(\ell_{\mathsf{app}})$
  $a_{\mathsf{to}} \leftarrow \mathsf{ExtractToAddress}(\ell_{\mathsf{app}})$; $\mathsf{ip} \leftarrow \mathsf{ExtractIP}(\ell_{\mathsf{net}})$
  $\mathcal{B}(r_i) \leftarrow$ preceding read burst of r_i
  $\psi_{\mathsf{entity}} \leftarrow [f(a_{\mathsf{from}}) \in S_{\mathsf{SDN}}]$
  $\psi_{\mathsf{juris}} \leftarrow [\mathsf{GeoLocate}(\mathsf{ip}) \in J_{\mathsf{OFAC}}]$
  $\psi_{\mathsf{contract}} \leftarrow [a_{\mathsf{to}} \in S_{\mathsf{SC}} \ \vee \ \exists r' \in \mathcal{B}(r_i) : \mathsf{ExtractToAddress}(r') \in S_{\mathsf{SC}}]$
  if $\psi_{\mathsf{entity}} \vee \psi_{\mathsf{juris}} \vee \psi_{\mathsf{contract}}$ **then** invoke C_{drop} on r_i
  else forward r_i unchanged
  end if
end for

Algorithm 5 Sandwich MEV construction

for each visible swap r with parameters $(\Delta_{\mathsf{in}}, \Delta_{\mathsf{out}}^{\min})$ **do**
  $b(f) \leftarrow \mathsf{Out}(r_X, r_Y, f, \varphi)$
  $y_v(f) \leftarrow \mathsf{Out}(r_X + f, r_Y - b(f), \Delta_{\mathsf{in}}, \varphi)$
  $f^* \leftarrow \max\{f \geq 0 : y_v(f) \geq \Delta_{\mathsf{out}}^{\min}\}$
  $b^* \leftarrow b(f^*)$
  $x_B \leftarrow \mathsf{Out}(r_Y - b^* - y_v(f^*), r_X + f^* + \Delta_{\mathsf{in}}, b^*, \varphi)$
  $\Pi^*(r) \leftarrow x_B - f^* - \mathsf{cost}_{\mathsf{incl}}$
  if $\Pi^*(r) > 0$ **then**
    construct front-run, back-run transactions of sizes f^*, b^*
    invoke $C_{\mathsf{delay}}/C_{\mathsf{drop}}$ on victim propagation
    bundle (tx_F, tx_V, tx_B) via C_{inject} and route via C_{route}
  end if
end for