

Consensus-level AML Enforcement for Compliant-by-Design Ledger Ecosystems

Mahmoud Shehadeh*, Panagiotis Michalopoulos*, Andreas Veneris*[†], Ronaldo Luo*

* Department of Electrical and Computer Engineering, University of Toronto

[†] Department of Computer Science, and Munk School of Global Affairs & Public Policy, University of Toronto

mahmoud.shehadeh@utoronto.ca, p.michalopoulos@mail.utoronto.ca, veneris@eecg.toronto.edu, ronaldo.luo@mail.utoronto.ca

Abstract—In recent years, blockchain networks have enabled the development of Decentralized Finance (DeFi) in which value transfers occur without centralized intermediaries. In traditional financial systems, such intermediaries have long acted as deputies to well established Anti-Money Laundering and Combating the Financing of Terrorism (AML/CFT) compliance measures. In contrast, by construction DeFi lacks such native AML/CFT mechanisms. As such, compliance is primarily enforced at centralized on- and off-ramps and is conducted reactively via third-party blockchain analytics services. In line with emerging global regulatory trends that seek to embed AML/CFT practices within modern DeFi environments, this paper presents a consensus-level framework that enables regulators to integrate these practices directly at the validator protocol layer. It models recent transactions as a graph and performs detection of common laundering patterns at the proposer/validator level for an Ethereum-like network architecture. Empirical results demonstrate a moderate overhead in exchange for proactive, protocol-native AML/CFT enforcement. In effect, this work not only complements existing AML/CFT DeFi practices but also opens new directions to address the multi-faceted challenges behind compliant decentralized system design.

Index Terms—Blockchain, Compliance by Design, AML/CFT, System Security for Decentralized Infrastructure

I. INTRODUCTION

In recent years, blockchain networks have enabled trustless programmable value transfers through smart contracts [1] without the use of centralized intermediaries. This underpins innovation in modern payment rails and forms the foundation of the rapidly growing Decentralized Finance (DeFi) ecosystems [2], [3]. In contrast, Traditional Finance (TradFi) systems operate within a two-tier model where fiat money is first issued by a central bank and subsequently distributed to the public by regulated intermediary Financial Institutions (FIs) such as private banks and other regulated payment institutions. Among other utilities, FIs have also conveniently served as deputies in enacting monetary policies and squashing tax evasion and other illicit activities by enforcing Anti-Money Laundering and Combating the Financing of Terrorism (AML/CFT), or AML for simplicity, compliance [4]. In comparison, blockchain systems lack embedded AML controls, a gap evidenced by industry reports which estimate that over \$158 billion in illicit on-chain activity was conducted in 2025 alone, or an equivalent of a shocking 145% increase from 2024 [5].

Modern AML compliance comprises of well-established domestic and international regulations and processes that intend to protect the financial system from illicit activity [6]. As noted earlier, in TradFi, these obligations mostly fall on FIs, which provide Know Your Customer (KYC) procedures and active cooperation with the authorities through transaction monitoring [7]. In most of the cases, this type of monitoring

involves *reactive* measures, where transactions are analyzed, flagged, and reported ex post. This is complemented by a limited number of *proactive* measures that operate in real time, such as blocking sanctioned and blacklisted entities, or enforcing risk-based payment thresholds [8].

Today, blockchains and DeFi protocols do not natively implement KYC or transaction-level AML controls. Instead, such compliance is primarily enforced at centralized on- and off-ramp points-of-entry mostly by Virtual Asset Service Providers (VASPs) [9] such as fiat-to-crypto exchanges. These agents leverage traditional KYC processes along with on-chain activity monitoring through reactive third-party analytics services [10], [11]. At the same time, recent regulatory developments, including the US GENIUS Act [12], the US Dept. of Treasury [13], EU’s MiCA [14], Canada’s Stablecoin Act [15], and Hong Kong’s Stablecoin Ordinance [16], signal a broader shift toward extending AML oversight to the blockchain ecosystem. This has prompted interest in *compliance-by-design* approaches [17], which integrate compliance objectives *within* the network mechanism(s) to enable proactive AML measures that complement legacy reactive ones.

One approach for proactive AML compliance is embedding checks directly into smart contracts. For instance, the issuers of the dominant USDT and USDC stablecoins implement address blacklists and freeze functions directly into their smart contracts enabling real-time prevention of token transfers involving illicit addresses [18], [19]. A recent study [20] extends this approach with programmable smart contract compliance rules and zero-knowledge (zk) proofs to verify users while preserving privacy. Although all these acts provide some essential gatekeeping, gas costs limit the complexity of on-chain AML rules that can be enforced beyond address blacklisting, basic rule-based checks, or single-transaction thresholds [20].

This paper is a departure from the techniques above as it introduces a validator-integrated proactive AML framework that embeds compliance *directly* at the consensus protocol level. The proposed framework models value-transferring transactions as edges in a dynamic transaction graph. Using this graph, validators perform proactive, real-time checks for illicit AML patterns, including the popular fan-in, fan-out, scatter-gather, and gather-scatter motifs [21] where transactions that violate these rules are marked as non-compliant. These AML patterns are guaranteed to be detected *deterministically* through a set of tunable thresholds, enabling regulatory domain experts to adapt the framework to different operational settings.

Unlike the smart contract centered methodologies, here the AML pattern checks are integrated into the block life-cycle, covering both block production and consensus, as follows. First, during block proposal, the validator evaluates transactions selected from the mempool (*i.e.*, pending transaction pool) against preset graph-based AML rules, excluding non-compliant transactions from the candidate block. Next, during consensus, validators independently augment their local transaction graphs with the proposed block's transactions. If the rules are violated, the block is deemed non-compliant and handled according to the consensus protocol's enforcement logic. This additional validation during consensus is necessary because block proposers cannot be assumed to be fully honest; for instance, recent empirical results for Ethereum confirm that proposers who claim to disregard transactions involving sanctioned "blacklisted" wallet addresses have still included them in blocks [22]. In our architecture, by enforcing these checks prior to block commitment, the framework prevents non-compliant transactions from being committed on-chain.

Notably, by construction, the framework deterministically detects *all* AML patterns for the set thresholds. The values of these thresholds are determined by the responsible regulatory authority and thus are not the topic of this work. As such, the paper instead focuses on evaluating the performance overhead introduced by the framework in the underlying blockchain system. To that end, we implement a prototype and evaluate it using a real-world workload of ERC-20 transactions. Our results show that the system sustains between 286 and 650 TPS, compared to 674 TPS with an unmodified validator, depending on the configured size of the transaction graph. Memory overhead remains modest, requiring only 16MB for a graph retaining transactions from the last 900 blocks, demonstrating that protocol-level AML is practically feasible.

The remainder of this paper is organized as follows: Section II provides a background of the relevant tools and concepts, along with related work. Section III details the proposed framework. Section IV presents the experimental results of our implementation, and Section V concludes the paper.

II. BACKGROUND AND RELATED WORK

A. System Assumptions

Given the wide adoption of Ethereum as the backbone of DeFi [23], we outline this work on an Ethereum-like architecture. In this environment, transactions are broadcast to the network where validators perform preliminary validity checks, such as verifying signatures, before admitting them into the local mempool. During block production, a designated proposer selects transactions from this pool to assemble a candidate block. This block is then broadcast to the network for consensus validation, where validators independently check the transactions to ensure they conform to the protocol's rules. A transaction is considered confirmed only when its block is accepted by the network and committed to the chain. Finally, for the sake of simplicity, we describe the protocol in terms of stablecoin transactions. Although it works with other cryptocurrency assets, their underlying price volatility

requires an oracle [24] to reevaluate AML thresholds in real time. Nevertheless, the protocol can be extended to handle this, as briefly argued later in the paper.

B. Money Laundering Patterns

Conceptually speaking, money laundering consists of three distinct stages: (i) *Placement*, the initial entry of illicit funds into the financial system; (ii) *Layering*, the obfuscation of funds through complex transaction chains across multiple accounts or intermediaries; (iii) *Integration*, the final re-entry of illicit funds into the legitimate economy [25].

In this paper, we focus on transaction-level AML for the layering stage; this is because the placement and integration stages of crypto-assets typically occur at on- and off-ramps through centralized exchanges. Additionally, the layering stage can encompass highly sophisticated decentralized tools such as mixers and cross-chain transfers [20]. Although we argue later in the paper how cross-chain transfer may be handled by our approach, the former cryptographic schemes that weaken and/or sever transaction linkability between transaction sources and destinations are not handled by the work here.

Specifically, we consider transaction-level patterns arising from the movement of funds across multiple wallets, which form identifiable typologies in transaction graphs. The selection of these typologies is based on regulatory guidance and recent institutional research. The Financial Action Task Force (FATF), the international standard-setting body for AML/CFT, identifies the structured transaction patterns used in this work as *significant* red flag indicators associated with the layering stage of money laundering [21]. Similarly, the Bank for International Settlements (BIS) further recognized and modeled these patterns as prominent financial crime typologies in Project Hertha [26]. We acknowledge that these structural patterns are neither exhaustive nor exclusively associated with illicit activity, as legitimate behaviour may produce the same patterns, a limitation we discuss later in the paper.

In more detail, based on the FATF guidance and BIS research, here we focus on the following AML typologies:

- **Fan-In** (Many-to-One): Multiple sources send funds to a single destination.
- **Fan-Out** (One-to-Many): A single source disperses funds to multiple destinations.
- **Scatter-Gather**: Funds from a single source are split through multiple intermediaries (One-to-Many) before consolidating at a final destination (Many-to-One).
- **Gather-Scatter**: Funds from multiple sources converge at a central intermediary (Many-to-One) before dispersing to multiple destinations (One-to-Many).

C. Existing AML/CFT Compliance Approaches

At the global level, the FATF acts as the primary standard setter for AML regulation, issuing recommendations that form the basis of national regulatory frameworks. These frameworks impose a range of obligations on regulated entities (*e.g.*, VASPs), including licensing requirements, the implementation

of customer due diligence and KYC measures, ongoing transaction monitoring and screening, and the retention of records. Regulated entities must assess their exposure to various risk factors and apply mitigation measures that are proportionate to the level of risk identified [27]. This *risk-based approach* relies on evaluation processes informed by multiple indicators, including individual transaction amounts as well as cumulative metrics calculated over defined time windows.

Regarding blockchain networks, various AML compliance solutions have been proposed that operate off-chain, at the smart contract level, at the validator/sequencer level, or through more comprehensive compliance frameworks:

Off-Chain AML Analytics. There is extensive prior work on AML methodologies that operate off-chain using on-chain data. Common techniques include transaction graph analysis [28], [29], anomaly detection [30], [31], and machine learning models to identify suspicious patterns [32], [33]. These systems are fundamentally reactive as they rely on centralized off-chain data that offer ex-post forensic insights.

Compliance-Enabled Smart Contracts. Liao et al. [34] extend the USDC smart contract with a transaction-proximity compliance flag that reflects a wallet’s distance to centralized exchanges, where addresses closer to centralized exchanges exhibit lower rates of illicit activity. This flag can be exposed to on-chain protocols or off-chain services, allowing them to calibrate risk scoring according to policy and risk tolerance. Further, the ERC-1400 and ERC-3643 standards provide compliance-aware tokens for regulated assets by embedding identity verification and/or rule-based transfer restrictions directly into smart contracts [35], [36]. In contrast to the proposed work, these approaches operate at the application layer and rely on contract-level enforcement, embedding compliance logic in smart contracts rather than at the protocol layer.

Sequencer and Mempool-Level Security. Zircuit [37] introduces a sequencer-level security protocol that integrates transaction screening directly into the rollup sequencer. Transactions are analyzed for malicious or policy-violating behaviour before block inclusion. However, this approach assumes a centralized sequencer model, which differs from the decentralized validator-based setting presented in this work. FrontDef [38] proposes a mempool-level security system that monitors incoming transactions to detect potentially malicious flash loan attacks. Upon detection, FrontDef can assemble transactions to front-run the attack and recover funds back to their rightful owner.

Privacy-Preserving Compliance Frameworks. RegKYC [39] is a privacy-preserving Attribute-Based Access Control (ABAC) framework that reconciles KYC/AML regulatory requirements with user privacy. Users are issued zk-enabled KYC credentials containing verifiable attributes, enabling service providers to enforce compliance policies while still supporting authorized deanonymization. Similarly, zkMixer [40] is a cryptocurrency mixer that validates deposits before entering the mixer pool. Validation occurs either via smart contract checks against predefined rules or via

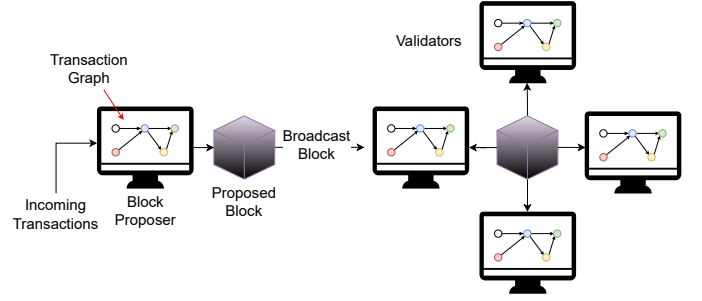


Fig. 1. Overview of the proposed framework

verifier-based approval, during which funds are held under a fixed time lock or indefinite hold. Verifiers then vote on whether deposits proceed, are refunded, or confiscated.

Admittedly, AML in blockchain networks remains a multifaceted challenge. Our work complements the aforementioned approaches by moving enforcement to the validator layer, enabling proactive real-time protocol-level defense against illicit transactions before their inclusion in the blockchain. To the best of our knowledge, this is the first work to integrate AML enforcement directly into the validator consensus protocol.

III. AML/CFT DETECTION FRAMEWORK

This section describes how AML/CFT is incorporated into the block production and consensus validation workflow. As shown in Fig. 1, all validators, including the block proposers, maintain a transaction graph in memory. This graph represents on-chain value transfer flows and is consistent across all validators. In the following subsections we describe the protocol actions when a new block is considered for addition to the chain at height h . During block proposal, the proposer uses the graph to evaluate candidate transactions from the mempool against predefined AML patterns. The resulting block is then broadcast and independently verified by the other validators by checking it against the same graph and AML patterns.

A. Transaction Graph

We represent on-chain activity as a directed multigraph $G = (V, E)$, where nodes represent on-chain addresses and directed edges $(s, r) \in E$ represent value-transferring transactions from sender s to receiver r . Only transactions that successfully execute are included in the graph. Transactions that fail or are reverted (e.g., due to insufficient balance or contract errors) do not create an edge, since no actual transfer of value occurs. The graph is *dynamic* in the sense that its structure changes as new transactions are added, and multiple transactions between the same two addresses are possible, i.e., G can be a multigraph. Each such transaction is denoted as $tx = (s, r, a, h)$, where a is the transferred amount, and h is the block height at which the transaction was included. Each edge is thus associated with a tuple (a, h) . Finally, the graph is bounded in the sense that it provides a global view of the network containing only the transactions within a sliding window of the w most recent validated blocks.

At the start of compiling the block at height h , the graph, denoted as $G_{\text{global}} = (V_{\text{global}}, E_{\text{global}})$, contains the transactions

TABLE I
AML DETECTION THRESHOLD PARAMETERS

Pattern	Parameter	Description
Fan-In	θ_{in}^{count}	Number of sources sending funds to the receiver
	θ_{in}^{sum}	Total amount received by the receiver from all sources
Fan-Out	θ_{out}^{count}	Number of recipients receiving funds from the sender
	θ_{out}^{sum}	Total amount sent by the sender to all recipients
Gather-Scatter (GS)	θ_{gs}^{count}	Number of sources sending to the sender and recipients receiving from it
	θ_{gs}^{sum}	Amount passing from multiple sources through the sender to recipients
Scatter-Gather (SG)	θ_{sg}^{count}	Number of intermediaries a source uses to reach the receiver
	θ_{sg}^{sum}	Amount sent from a source through multiple intermediaries to the receiver

from the committed blocks in the block range $[h - w, h - 1]$. Let $\{tx_1, tx_2, \dots, tx_n\}$ be the transactions proposed for inclusion in the current block. Transaction evaluation proceeds sequentially by incrementally extending the graph, through a collection of transient graphs $G^{(i)}$ ($i \in [0, n]$) starting from

$$G^{(0)} = G_{\text{global}}.$$

For each transaction tx_i , $i \geq 1$, the graph is updated as

$$G^{(i)} = \begin{cases} G^{(i-1)} \cup \{tx_i\}, & \text{if } tx_i \text{ is compliant,} \\ G^{(i-1)}, & \text{otherwise.} \end{cases}$$

This update mechanism ensures that only compliant transactions contribute to the graph's state for subsequent transaction evaluations. After the block is verified and committed to the chain at block height h , G_{global} is updated to $G^{(n)}$ and subsequently pruned to remove transactions from blocks that fall outside the sliding window.

B. AML Pattern Detection

When transaction tx_i , where $i \in [1, n]$, is examined for inclusion in the block at height h , it is evaluated for compliance against the information within $G^{(i-1)}$ using the AML patterns described in Section II-B and the threshold parameters listed in Table I. These threshold parameters are grounded in *established AML practices*. FATF guidance identifies transaction amounts and the movement of funds across multiple accounts as relevant red flag indicators [21]. We reflect these indicators through amount thresholds that quantify the value transferred through each pattern and count thresholds that capture the structure of the AML pattern through the number of distinct sources, recipients, or intermediaries involved. Thus, these parameters provide a means to (i) assess whether a transaction exhibits potentially illicit behaviour based on the transferred amount and the number of participating accounts, and (ii) adapt the framework to different regulatory settings. Potential updates to these thresholds, as determined by the regulatory authority, take effect at a predetermined future block height, ensuring synchronization across validators. This follows an approach analogous to Ethereum protocol upgrades.

The AML patterns are categorized as *sender-centric* or *receiver-centric* based on which participant in a transaction

is being evaluated. The sender-centric logic assesses whether the sender exhibits Fan-Out fund dispersion or acts as a central hub in Gather-Scatter flows. In contrast, the receiver-centric logic evaluates whether the receiver consolidates funds from multiple sources in Fan-In patterns or receives funds that have been dispersed through multiple intermediaries in a Scatter-Gather sequence. The sender-centric logic, detailed in Algorithm 1, evaluates the patterns as follows:

a) *Fan-Out*: The Fan-Out pattern is evaluated by first retrieving the outgoingEdges from s , including the current transaction (lines 2–3). The algorithm then computes the total outgoingAmt and the number of distinct recipients recipientsCount receiving funds from s (lines 4–5). If either recipientsCount exceeds θ_{out}^{count} or outgoingAmt exceeds θ_{out}^{sum} , the transaction is flagged as illicit (lines 6–7).

b) *Gather-Scatter*: The Gather-Scatter pattern is evaluated by tracing funds originating from multiple sources and passing through the sender s . The algorithm first retrieves the incomingEdges to s and determines the number of distinct sources sourcesCount sending funds to s (lines 9–10). If both sourcesCount and recipientsCount exceed θ_{gs}^{count} (line 11), the algorithm computes incomingAmt by summing the amounts of incomingEdges and determines the amount passing through s as the minimum of incomingAmt and outgoingAmt (line 12). If this amount exceeds θ_{gs}^{sum} , the transaction is flagged as illicit (lines 13–14).

Algorithm 1 Sender-Centric AML Evaluation

```

1: procedure EVALUATESENDER( $s, r, a, h$ )
   $\triangleright$  Fan-Out
2:   outgoingEdges  $\leftarrow$  GetOutgoingEdges( $s$ )
3:   outgoingEdges.append( $((s, r, a, h))$ )
4:   outgoingAmt  $\leftarrow$  Sum(outgoingEdges)
5:   recipientsCount  $\leftarrow$  CountRecipients(outgoingEdges)
6:   if recipientsCount >  $\theta_{out}^{count}$  or outgoingAmt >  $\theta_{out}^{sum}$ 
     then
7:     return ILLICIT
8:
   $\triangleright$  Gather-Scatter
9:   incomingEdges  $\leftarrow$  GetIncomingEdges( $s$ )
10:  sourcesCount  $\leftarrow$  CountSources(incomingEdges)
11:  if sourcesCount >  $\theta_{gs}^{count}$  and recipientsCount >  $\theta_{gs}^{count}$  then
12:    incomingAmt  $\leftarrow$  Sum(incomingEdges)
13:    if min(incomingAmt, outgoingAmt) >  $\theta_{gs}^{sum}$  then
14:      return ILLICIT
15:  return VALID

```

Next, Algorithm 2 evaluates receiver-centric patterns:

a) *Fan-In*: The Fan-In pattern is evaluated by first retrieving the incomingEdges to the receiver r , including the current transaction (lines 2–3). The algorithm then computes the total incomingAmt and the number of distinct sources sourcesCount associated with these edges (lines 4–5). If either sourcesCount exceeds θ_{in}^{count} or incomingAmt exceeds θ_{in}^{sum} , the transaction is flagged as illicit (lines 6–7).

b) *Scatter-Gather*: The Scatter-Gather pattern flags funds that are split in smaller amounts at a source node and re-

consolidated at receiver r through different intermediaries. For each intermediary node i , the algorithm: (i) computes the total amount sent from i to r , denoted as amtToReceiver (line 11); and (ii) retrieves the incoming edges to i occurring at or before the latest $i \rightarrow r$ transfer, stored in sourceEdges (line 12).

For each source src in these edges, it computes the total amount sent from src to i as amtToInter (line 14), and calculates the minimum of amtToInter and amtToReceiver as vol (line 15). This represents the amount from src that passed through i to reach r . The volume vol is added to $\text{sourceToVolume}[\text{src}]$, and i is recorded in $\text{sourceToInter}[\text{src}]$ to track distinct intermediaries used by each src (lines 16–17).

Finally, for each src , if the count of distinct intermediaries in $\text{sourceToInter}[\text{src}]$ exceeds $\theta_{\text{sg}}^{\text{count}}$ and the aggregated volume in $\text{sourceToVolume}[\text{src}]$ exceeds $\theta_{\text{sg}}^{\text{sum}}$, the transaction is flagged as illicit (lines 18–21).

Algorithm 2 Receiver-Centric AML Evaluation

```

1: procedure EVALUATERECEIVER( $s, r, a, h$ )
  ▷ Fan-In
2:    $\text{incomingEdges} \leftarrow \text{GetIncomingEdges}(r)$ 
3:    $\text{incomingEdges.append}((s, r, a, h))$ 
4:    $\text{incomingAmt} \leftarrow \text{Sum}(\text{incomingEdges})$ 
5:    $\text{sourcesCount} \leftarrow \text{CountSources}(\text{incomingEdges})$ 
6:   if  $\text{sourcesCount} > \theta_{\text{in}}^{\text{count}}$  or  $\text{incomingAmt} > \theta_{\text{in}}^{\text{sum}}$  then
7:     return ILLICIT
8:
  ▷ Scatter-Gather
9:   Init mappings  $\text{sourceToInter}$  and  $\text{sourceToVolume}$ 
10:  for each intermediary  $i$  that sent to  $r$  do
11:     $\text{amtToReceiver} \leftarrow$  total amount  $i$  sent to  $r$ 
12:     $\text{sourceEdges} \leftarrow$  incoming edges to  $i$  at block
      heights  $\leq$  latest block height of  $i$  to  $r$  transfers
13:    for each source  $\text{src}$  in  $\text{sourceEdges}$  do
14:       $\text{amtToInter} \leftarrow$  total amount  $\text{src}$  sent to  $i$ 
15:       $\text{vol} \leftarrow \min(\text{amtToInter}, \text{amtToReceiver})$ 
16:       $\text{sourceToInter}[\text{src}].\text{add}(i)$ 
17:       $\text{sourceToVolume}[\text{src}] += \text{vol}$ 
18:  for each source  $\text{src} \in \text{keys}(\text{sourceToInter})$  do
19:    if  $\text{len}(\text{sourceToInter}[\text{src}]) > \theta_{\text{sg}}^{\text{count}}$  then
20:      if  $\text{sourceToVolume}[\text{src}] > \theta_{\text{sg}}^{\text{sum}}$  then
21:        return ILLICIT
22:  return VALID

```

C. The Consensus Protocol

In this subsection we describe how a new block at height h is proposed and validated following AML pattern detection. We also outline how the various information is utilized to update the transaction graph as new blocks are committed.

Proposer Block Production: During block production, the proposer selects n candidate transactions, $\{tx_1, \dots, tx_n\}$, from its mempool and evaluates them sequentially in their order of selection. When examining inclusion of transaction tx_i the proposer evaluates it, using Algorithms 1 and 2, against the current graph state $G^{(i-1)}$, which consists of G_{global} at height $h - 1$ augmented by all previously successfully evaluated transactions $\{tx_1, \dots, tx_{i-1}\}$ in the said candidate

block. If tx_i is compliant and does not trigger an AML pattern, it is included in the block and the graph is updated to $G^{(i)} = G^{(i-1)} \cup \{tx_i\}$. If tx_i triggers an AML pattern, it is rejected and excluded from the candidate block, with the graph remaining unchanged (*i.e.*, $G^{(i)} = G^{(i-1)}$). In case an excluded transaction creates a nonce gap, subsequent transactions from the same sender remain pending until the sender submits an admissible replacement for the missing nonce. Once the transaction evaluation process is complete, the candidate block is broadcast to the network.

Consensus Validation: Upon receiving the proposed block, validators independently verify its validity as part of the consensus protocol. The proposed framework extends standard protocol checks (*e.g.*, valid signatures, hashes, etc.) with the AML pattern checks to ensure the block is AML-compliant. Validators evaluate transactions $\{tx_1, \dots, tx_n\}$ of the block in the same manner as the proposer starting from $G^{(0)} = G_{\text{global}}$. Each tx_i ($i \in [1, n]$) is first evaluated against $G^{(i-1)}$ and if it is valid, it is added to the graph as part of $G^{(i)}$; otherwise, validators omit its effects (*i.e.*, $G^{(i)} = G^{(i-1)}$). This process continues for the remaining transactions to assess their individual compliance status. Because validators start from the same G_{global} and evaluate transactions in the same order, all honest validators arrive at identical conclusions regarding which transactions trigger AML patterns. The final status of the block and whether it is committed to the chain is subsequently determined by the consensus protocol's rules as elaborated at the end of this section.

Transaction Graph Management: Once a valid, compliant block has been accepted by the consensus protocol, all validators incorporate the successful, legitimate transactions from that block at height h into their G_{global} , thereby ensuring a consistent G_{global} across validators. This process is detailed in Algorithm 3. For each transaction (s, r, a, h) , the addresses are added to V_{global} if not already present, and a directed edge is created in E_{global} (lines 4–5). The validator records an identifier for each edge in a map edgesPerBlock , keyed by the block height h (line 6). This enables efficient block-level edge lookup without traversing the entire graph.

To keep G_{global} bounded to the last w blocks, block heights are appended to a FIFO queue, blockQueue (line 7). When the oldest block height falls outside the range of the sliding window (line 8), all transactions associated with that block are removed from G_{global} . This is done by retrieving the block's edges via edgesPerBlock and removing them from E_{global} (lines 9–11). Any nodes that become isolated (*i.e.*, have a degree of zero) after edge removal are also removed from V_{global} (line 12). Finally, the entry for the removed block is deleted from edgesPerBlock (line 13). This ensures that G_{global} contains only transactions relevant for AML evaluation.

Chain Reorganizations: A chain reorganization occurs when a validator switches from its currently accepted chain to a newly adopted chain, for example due to a longer chain emerging or temporary forks. To maintain an accurate transaction graph, edges corresponding to blocks from the previous chain

Algorithm 3 Transaction Graph Management

```
1: procedure UPDATEGRAPH( $h$ , transactions)
2:   edgesPerBlock[ $h$ ]  $\leftarrow$  empty list
3:   for each  $(s, r, a, h) \in$  transactions do
4:     Add  $s$  and  $r$  to  $V_{\text{global}}$  if not present
5:     Add edge  $(s, r, a, h)$  to  $E_{\text{global}}$ 
6:     Store edge ID in edgesPerBlock[ $h$ ]
7:   blockQueue.append( $h$ )
8:   if blockQueue.front()  $\leq h - w$  then
9:     oldHeight  $\leftarrow$  blockQueue.pop()
10:    for each edge ID in edgesPerBlock[oldHeight] do
11:      Remove corresponding edge from  $E_{\text{global}}$ 
12:    Remove isolated nodes from  $V_{\text{global}}$ 
13:    delete edgesPerBlock[oldHeight]
```

that are not part of the new chain are removed from the graph, and any nodes that become isolated are deleted. As the blocks of the new chain are validated and added sequentially, their transactions are incorporated into G_{global} . This process ensures that G_{global} accurately reflects the chain state.

D. AML within a Cross-Chain Environment

The protocol above can be adapted to work in a cross-chain setting where validators participate on multiple chains by simultaneously running different virtual machines allowing seamless communication between smart contracts across chains [41]. These validators can maintain a unified transaction graph that includes both intra-chain and cross-chain transfers, providing a holistic view of value flows across the participating chains. Notably, this requires that all validators collectively operate across these chains, ensuring they maintain an identical and consistent transaction graph. However, in bridge-based cross-chain interactions, transactions entering a bridge on one chain have no on-chain link to their corresponding outputs on the destination chain. Consequently, the transaction graph cannot directly trace these cross-chain transfers, preventing consistent tracking of value transfers. Nonetheless, if the bridge interaction provides a way to identify the sender on the source chain and the receiver on the destination chain, a mapping correspondence can be established, allowing validators to capture the bridged transfer in the transaction graph.

E. Discussion & Directions

Evidently, the integration of AML checks directly into the blockchain consensus protocol raises questions on *how* validators should handle proposed blocks that contain provenly illicit transactions. In typical scenarios today, the consensus mechanism of blockchain networks enforces rudimentary “agreed upon” protocol rules where blocks containing invalid transactions (such as those with incorrect signatures or double-spends) are rejected. By analogy, the presence of an illicit transaction could similarly render the entire block invalid and validators should reject such a block during consensus validation. To discourage the repeated inclusion of illicit transactions, which could create the potential for denial-of-service attacks, block proposers could be subject to penalties such as stake slashing. This would provide a strong incentive

for proposers to include only compliant transactions, since a violation would result in block rejection and potential slashing.

Alternatively, to address potential impacts to throughput caused by block rejections, consensus mechanisms could be amended to support more flexible handling of illicit transactions – *e.g.*, validators could first identify all illicit transactions in a proposed block, then either nullify the offending transactions while preserving the remaining compliant ones, or filter them out to create a modified block. Subsequently, an additional expedited round of consensus focused on validating the revised block could be used. In regulated environments, another avenue to reduce disruption to legitimate activity is to conditionally allow flagged transactions to proceed while recording their AML status as suspicious, thereby triggering subsequent review and any necessary remedial action.

Another consideration is the value of the window size w . Increasing w allows for more historical transaction data to be considered during AML, reducing the impact of evasion attempts by malicious proposers, such as short-term transaction reordering or transaction spreading over time. However, it directly increases the memory required to store the transaction graph. It also influences the computational overhead in three key stages of the protocol: evaluating each transaction during block production, validating the block during consensus, and updating the transaction graph after a block is added. These effects are further discussed in the experimental section that follows. Nevertheless, in modern AML TradFi settings, w could correspond to months or even years of transaction history, which may challenge the approach as presented here. Therefore, techniques such as *incremental graph summarization* [42] or machine learning models that leverage graph embeddings (*e.g.*, *temporal graph networks* [43]) could be utilized as they do not require storing the full transaction history. Because such heuristics, but also machine learning inference, are inherently stochastic and non-deterministic processes (thus leading to different validators reaching different conclusions about transactions of the same block), developing approaches to ensure determinism in the proposed setting sets interesting paths for future work.

Finally, as noted earlier, this framework was described above in terms of stablecoins. To extend it to other cryptocurrencies with volatile values, a deterministic price source, such as an oracle, would be required [24]. Specifically, both proposers and validators could use the value provided by the oracle to convert the price of a crypto asset into a fiat-denominated value to be stored as a universal parameter a in the transaction graph.

IV. EVALUATION

In this section, we evaluate the performance impact of integrating the proposed AML framework into an Ethereum client, focusing on how varying the window size w affects performance. We measure transaction throughput, the memory overhead of maintaining the transaction graph, and latency overheads at each stage of the protocol: block production, con-

sensus validation, and transaction graph management during block commitment, in an effort to infer its practicality.

A. Implementation

We implement the proposed framework by modifying Reth, a high-performance Ethereum execution client written in Rust [44]. In particular, we integrate AML pattern detection into the modules responsible for transaction selection during block production and block validation during consensus. In addition, we alter the process of adding a block to the chain to also update the transaction graph G_{global} .

B. Experimental Setup

The experiments are run on a machine with an AMD Ryzen 7 7745HX processor, 32GB RAM, and 1TB SSD storage. We run our implementation using Reth v1.9.3, with the unmodified (vanilla) client serving as the baseline for throughput measurements. In order to isolate the computational overhead introduced by the AML framework, we run experiments with a single node, so that blocks can be instantly mined and network effects will be negligible. By construction, in a multi-node setting, this per-node cost is incurred by all validators *in parallel*. Therefore, the single-node setup is indicative of the system overhead that would also be introduced in a full multi-node blockchain system.

We construct an experimental workload by collecting a trace of 200,000 ERC-20 transfer transactions involving the USDC token from the Ethereum mainnet, sampled from blocks 24,177,261 to 24,179,390. This consists of 46,922 unique senders and 55,964 unique recipients. The transactions are replayed in their original block order and submitted to the mempool in block-sized batches, ensuring that transactions from the same sampled block are packed together.

False positives and negatives: False positives (FPs) occur when a detected AML pattern corresponds to legitimate transaction activity, whereas false negatives (FNs) occur when illicit activity is not captured by the configured patterns. AML patterns are detected according to (i) the graph structure, and (ii) the set parameterization, *i.e.*, the transferred amounts and the number of participating accounts. As noted earlier, for a given set of threshold values, the framework *deterministically* detects *all* instances of the graph structures that correspond to the configured AML patterns through its underlying design. Whether those detected instances correspond to legitimate or illicit activity depends on the selected threshold values — parameters configured by regulatory/enforcement authorities based on historical transaction data. Consequently, the occurrence of FPs and FNs depends on the configured thresholds rather than on the framework itself. As those thresholds are set by authorities based on proprietary information, we cannot evaluate detection outcomes. We note that further investigation into how FP and FN rates can be determined without such data is needed. Instead, we focus on evaluating the system-level performance and the overhead it introduces to detect *all* configured AML patterns based on our set thresholds.

To this end, we set each of the thresholds $\theta_{\text{in}}^{\text{count}}$, $\theta_{\text{out}}^{\text{count}}$, $\theta_{\text{gs}}^{\text{count}}$, and $\theta_{\text{sg}}^{\text{count}}$ to the value $2^{64} - 1$ and those of the parameters

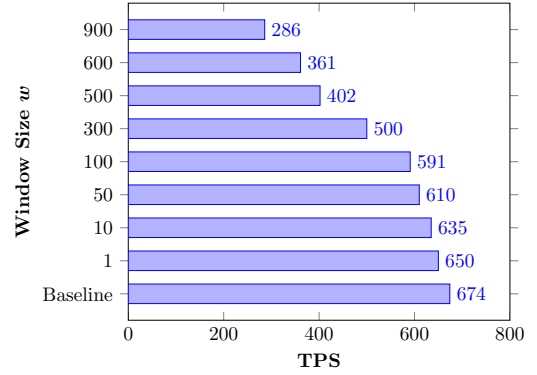


Fig. 2. TPS for various values of window w

TABLE II
TPS ACROSS DIFFERENT AML PATTERNS

Window Size w	AML Patterns		
	Fan-In & Fan-Out	Sender-Only	Receiver-Only
1	650	655	650
10	634	652	647
50	628	631	630
100	624	626	609
300	567	574	563
500	493	498	504
600	466	463	499
900	376	381	436

$\theta_{\text{in}}^{\text{sum}}$, $\theta_{\text{out}}^{\text{sum}}$, $\theta_{\text{gs}}^{\text{sum}}$, and $\theta_{\text{sg}}^{\text{sum}}$ to the value of $2^{256} - 1$. This avoids triggering any AML patterns while stress-testing the system under worst-case conditions, ensuring that the full suite of AML patterns is evaluated for each transaction.

We deploy an ERC-20 contract on the single-node and synthesize sender and receiver accounts according to the USDC transaction trace. All sender accounts are then funded with sufficient token balances and ETH to execute the transfers. Subsequently, the ERC-20 transfers are replayed using these generated accounts, following the transaction trace.

C. Experimental Results

Fig. 2 illustrates the impact on TPS as w increases, with the baseline corresponding to the unmodified client operating without any AML detection. For reference, Ethereum generates a block approximately every 12 seconds, so, for instance, $w = 300$ corresponds to roughly 1 hour of transaction history. As w increases, we observe a gradual decrease in TPS. This is because the graph retains more transactions, resulting in increased computational overhead for evaluating each new transaction against the graph.

Table II presents the TPS when evaluating a subset of AML patterns, namely Fan-In & Fan-Out, Sender-Only, and Receiver-Only patterns. Overall, TPS is higher compared to when evaluating all patterns, with differences becoming more pronounced for larger window sizes. This demonstrates that high TPS can still be achieved with larger w , though at the cost of the comprehensiveness of the AML patterns evaluated.

We measure the memory overhead of the transaction graph, including the auxiliary data structures required for graph maintenance, in Fig. 3. For small window sizes ($w \leq 100$), the memory overhead remains modest, staying below 3MB.

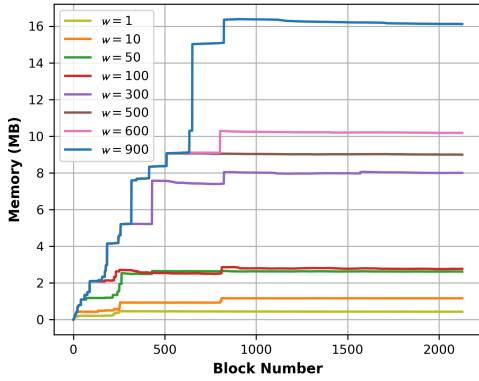


Fig. 3. Memory overhead for various values of w

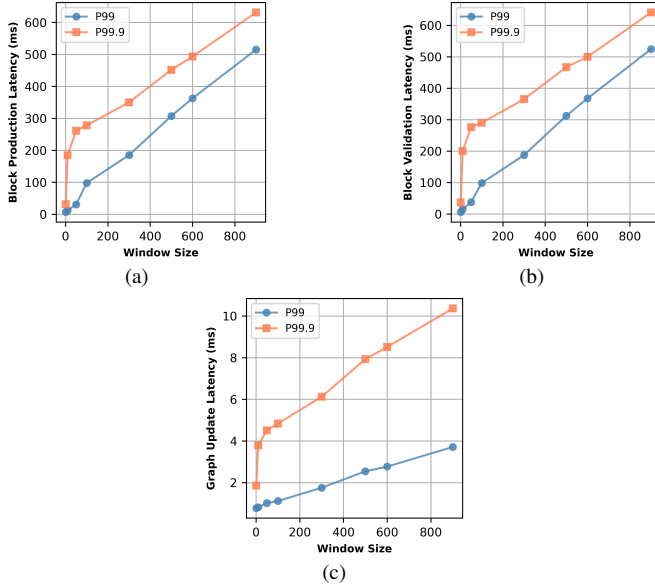


Fig. 4. Latency at each protocol stage

The memory overhead becomes more pronounced for larger window sizes, particularly for $w \geq 300$; however, even at $w = 900$, the total memory usage is approximately 16MB, which remains rather manageable on modern machines. The stepwise increases in memory usage observed in the figure correspond to heap allocations triggered as additional transactions are incorporated into the graph. Once sufficient memory has been allocated to accommodate the full size of the graph, memory usage plateaus for each w . This behaviour shows that the transaction graph does not grow unbounded over time, but instead stabilizes according to the window size.

We also examine the latency associated with each stage of the protocol in Fig. 4. To characterize worst-case behaviour, we report the p99 and p99.9 percentiles, which capture tail latency rather than average or median values. This is particularly important to account for blocks containing a large number of transactions or transactions exhibiting high interaction patterns, such as smart contract invocations or hot-wallet activity. As shown in Fig. 4a and Fig. 4b, the overhead introduced during block production and consensus block validation is nearly identical, as both stages check the same AML patterns over the same set of transactions in the

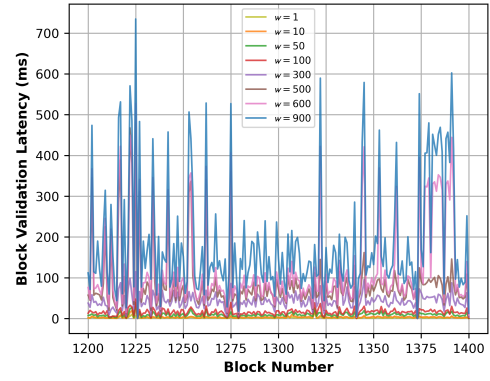


Fig. 5. Consensus block validation latency

same order. Across all of Fig. 4, we observe a clear upward trend in latency as the window size w increases. Notably, the transaction graph update latency in Fig. 4c grows only slightly, with a p99.9 remaining below 11ms, whereas block production and block validation exhibit much larger increases, reaching p99 of 520ms and p99.9 of 630ms.

Fig. 5 shows the block validation latency during consensus measured across consecutive blocks. To better illustrate trends, we focus on a representative snapshot of 200 blocks and select a period after memory usage has plateaued. This ensures the transaction graph has reached a steady state, as block 1200 exceeds the maximum window size w considered. Evidently, larger window sizes consistently result in higher block validation latency as more transaction history is retained; a similar trend is to be expected when tracking additional token types, as this would increase the number of nodes and transaction edges maintained in the graph. We also observe latency spikes across all window sizes. These spikes typically correspond to blocks containing a large number of ERC-20 transactions, particularly those involving high-activity smart contracts or externally owned accounts with elevated transaction volumes, such as hot wallets operated by centralized exchanges. The effect of such blocks becomes more pronounced at larger window sizes. For example, at $w = 900$, block validation latency occasionally exceeds 500ms. Despite these spikes, the observed validation latencies remain within acceptable bounds for a consensus protocol such as Ethereum's. Even in the worst case, block validation completes within the approximately 4-second window available for validators to verify and attest to the proposed block.

V. CONCLUSION

The process of AML/CFT regulatory compliance in permissionless and permissioned blockchain networks remains a persistent challenge; today, it is often addressed through reactive measures or by TradFi-like ones that do not fully capture the network effects and innovation behind decentralization. This work demonstrates that integrating said detection rules directly into the validator protocol offers a promising direction for proactive enforcement that enhances existing practices while embracing the spirit of compliant-by-design Web 3.0 systems. Ultimately, the results here show that integrating AML/CFT

detection into the consensus protocol is both practical and effective in terms of system latency and memory overhead.

VI. ACKNOWLEDGMENTS

The authors acknowledge various comments and insights by Dr. Keerthi Nelaturu and the anonymous reviewers that helped improve the content and presentation of this work.

REFERENCES

- [1] V. Buterin *et al.*, “A next-generation smart contract and decentralized application platform,” *white paper*, vol. 3, no. 37, pp. 2–1, 2014.
- [2] M. Aquilina, G. Cornelli, J. Frost, and L. Gambacorta, “Cryptocurrencies and decentralised finance: Functions and financial stability implications,” Bank for International Settlements, BIS Paper 156, Apr. 2025. [Online]. Available: <https://www.bis.org/publ/bppdf/bispap156.htm>
- [3] M. D. Maggio, *Blockchain, Crypto and DeFi: Bridging Finance and Technology*. Hoboken, NJ: John Wiley & Sons, 2024.
- [4] A. Y. Shehu, “Promoting financial sector stability through an effective AML/CFT regime,” *Journal of Money Laundering Control*, vol. 13, no. 2, pp. 139–154, 05 2010. [Online]. Available: <https://doi.org/10.1108/13685201011034087>
- [5] TRM Labs, “2026 Crypto Crime Report,” [Online]. Available: <https://www.trmlabs.com/reports-and-whitepapers/2026-crypto-crime-report>, Jan. 2026, (accessed Jul. 29, 2026).
- [6] M. Levi and P. Reuter, “Money laundering,” *Crime and justice*, vol. 34, no. 1, pp. 289–375, 2006.
- [7] N. Mugarura, “Customer due diligence (CDD) mandate and the propensity of its application as a global AML paradigm,” *Journal of Money Laundering Control*, vol. 17, no. 1, pp. 76–95, 01 2014. [Online]. Available: <https://doi.org/10.1108/JMLC-07-2013-0024>
- [8] M. E. Edge, P. R. F. Sampaio, and M. Choudhary, “Towards a proactive fraud management framework for financial data streams,” in *Third IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC 2007)*, 2007, pp. 55–64.
- [9] M. Barczentewicz, “Regulatory compliance for crypto by relying on on-ramping services,” April 2022. [Online]. Available: <https://ssrn.com/abstract=4068101>
- [10] Elliptic, “Elliptic – Blockchain Analytics and Crypto Compliance Solutions,” 2025, <https://www.elliptic.co/> (accessed Jul. 29, 2026).
- [11] Chainalysis, “Chainalysis – blockchain data platform,” 2025, <https://www.chainalysis.com/> (accessed Jul. 29, 2026).
- [12] U.S. Congress, “S.1582 - 119th Congress (2025–2026): GENIUS Act,” Congress.gov, July 2025, public Law No. 119-27. [Online]. Available: <https://www.congress.gov/bills/119th-congress/senate-bill/1582>
- [13] U.S. Department of the Treasury, “Request for Comment on Innovative Methods To Detect Illicit Activity Involving Digital Assets,” Federal Register, Aug. 2025, (accessed Jul. 29 2026). [Online]. Available: <https://www.federalregister.gov/documents/2025/08/18/2025-15697/request-for-comment-on-innovative-methods-to-detect-illicit-activity-involving-digital-assets>
- [14] European Union, “Regulation (EU) 2023/1114 of the European parliament and of the council of 31 May 2023 on markets in crypto-assets, and amending Regulations (EU) No 1093/2010 and (EU) No 1095/2010 and Directives 2013/36/EU and (EU) 2019/1937,” 2023.
- [15] Parliament of Canada, “Bill C-15: An Act to implement certain provisions of the budget tabled in Parliament on November 4, 2025 (Budget 2025 Implementation Act, No. 1), Part 5, Division 45 (Stablecoin Act),” 2025.
- [16] Hong Kong Special Administrative Region, “Stablecoins Ordinance (Cap. 656),” aug 2025.
- [17] N. Pocher and A. Veneris, “Privacy and transparency in CBDCs: A regulation-by-design AML/CFT scheme,” *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1776–1788, 2022.
- [18] Circle, “USDC Risk Factors,” circle.com. <https://www.circle.com/legal/usdc-risk-factors> (accessed Jul. 29, 2026).
- [19] Tether, “Legal,” <https://tether.to/en/legal> (accessed Jul. 29, 2026).
- [20] J. D. Duffie, O. Olowookere, and A. Veneris, “Comment in Response to the U.S. Department of the Treasury’s Advanced Notice of Proposed Rulemaking on the Guiding and Establishing National Innovation for U.S. Stablecoins (GENIUS) Act Implementation,” Nov. 2025. [Online]. Available: <https://ssrn.com/abstract=5692624>
- [21] Financial Action Task Force (FATF), “Virtual assets red flag indicators of money laundering and terrorist financing,” 2020. [Online]. Available: <https://www.fatf-gafi.org/content/dam/fatf-gafi/reports/Virtual-Assets-Red-Flag-Indicators.pdf>
- [22] L. Heimbach, L. Kiffer, C. F. Torres, and R. Wattenhofer, “Ethereum’s proposer-builder separation: Promises and realities,” 2023. [Online]. Available: <https://arxiv.org/abs/2305.19037>
- [23] DefiLlama, “Ethereum,” [defillama.com. https://defillama.com/chain/ethereum](https://defillama.com/chain/ethereum) (accessed Jul. 28, 2026).
- [24] A. Pasdar, Y. C. Lee, and Z. Dong, “Connect API with blockchain: A survey on blockchain oracle implementation,” *ACM Computing Surveys*, vol. 55, no. 10, pp. 1–39, 2023.
- [25] F. Schneider and U. Windischbauer, “Money laundering: some facts,” *European Journal of Law and Economics*, vol. 26, no. 3, pp. 387–404, Dec 2008. [Online]. Available: <https://doi.org/10.1007/s10657-008-9070-x>
- [26] BIS, “Project Hertha: Identifying financial crime patterns in a real-time retail payment systems,” [Online]. Available: <https://www.bis.org/publ/othp96.pdf>, Jun. 2025, (accessed Jul. 29, 2026).
- [27] Financial Action Task Force (FATF), “International standards on combating money laundering and the financing of terrorism & proliferation: The FATF recommendations,” 2012. [Online]. Available: <https://www.fatf-gafi.org/content/dam/fatf-gafi/recommendations/FATF%20Recommendations%202012.pdf>
- [28] D. Lin, J. Wu, Y. Yu, Q. Fu, Z. Zheng, and C. Yang, “DenseFlow: Spotting cryptocurrency money laundering in ethereum transaction graphs,” in *Proceedings of the ACM Web Conference 2024*, ser. WWW ’24, 2024, p. 4429–4438. [Online]. Available: <https://doi.org/10.1145/3589334.3645692>
- [29] K. Song, M. A. Dhraief, M. Xu, L. Cai, X. Chen, A. Mithal, and J. Chen, “Identifying money laundering subgraphs on the blockchain,” in *Proceedings of the 5th ACM International Conference on AI in Finance*, ser. ICAIF ’24, 2024, p. 195–203. [Online]. Available: <https://doi.org/10.1145/3677052.3698635>
- [30] T. Pham and S. Lee, “Anomaly detection in the bitcoin system-a network perspective,” *arXiv preprint arXiv:1611.03942*, 2016.
- [31] P. M. Monamo, V. Marivate, and B. Twala, “A multifaceted approach to bitcoin fraud detection: Global and local outliers,” in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2016, pp. 188–194.
- [32] S. Marasi and S. Ferretti, “Anti-money laundering in cryptocurrencies through graph neural networks: A comparative study,” in *2024 IEEE 21st Consumer Communications & Networking Conference (CCNC)*. IEEE, 2024, pp. 272–277.
- [33] C. Bellei, M. Xu, R. Phillips, T. Robinson, M. Weber, T. Kaler, C. E. Leiserson, Arvind, and J. Chen, “The shape of money laundering: Subgraph representation learning on the blockchain with the elliptic2 dataset,” 2024. [Online]. Available: <https://arxiv.org/abs/2404.19109>
- [34] G. Y. Liao, Z. Zeng, M. Belenkiy, and J. Hirshman, “Transaction Proximity: A graph-based approach to blockchain fraud prevention,” 2025. [Online]. Available: <https://arxiv.org/abs/2505.24284>
- [35] A. Dossa, P. Ruiz, F. Vogelsteller, and S. Gosselin, “ERC 1400: Security Token Standard,” [Online]. Available: <https://github.com/ethereum/EIPs/issues/1411>, Sep. 2018, (accessed Jul. 28, 2026).
- [36] J. Lebrun, T. Malghem, K. Thizy, L. Falempin, and A. Boudjemaa, “ERC-3643: T-REX - Token for Regulated EXchanges,” [Online]. Available: <https://eips.ethereum.org/EIPS/eip-3643>, Jul. 2021, (accessed Jul. 28, 2026).
- [37] M. Derka, J. Gorzny, D. Siqueira, D. Pellegrino, M. Guggenmos, and Z. Chen, “Sequencer level security,” 2024. [Online]. Available: <https://arxiv.org/abs/2405.01819>
- [38] X. Deng, Z. Zhao, S. M. Beillahi, H. Du, C. Minwalla, K. Nelaturu, A. Veneris, and F. Long, “A robust front-running methodology for malicious flash-loan DeFi attacks,” in *2023 IEEE International Conference on Decentralized Applications and Infrastructures*, 2023, pp. 38–47.
- [39] X. Xiong, M. Huth, and W. Knottenbelt, “RegKYC: Supporting privacy and compliance enforcement for KYC in blockchains,” in *2025 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2025, pp. 1–5.
- [40] T. Constantinides and J. Cartlidge, “zkMixer: A configurable zero-knowledge mixer with anti-money laundering consensus protocols,” in *2025 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, 2025, pp. 111–120.
- [41] V. Nekriach, S. M. Beillahi, C. Li, P. Li, M. Wu, A. Veneris, and F. Long, “HEMVM: A heterogeneous blockchain framework for interoperable virtual machines,” *Proc. ACM Program. Lang.*, vol. 9, no. OOPSLA2, Oct. 2025. [Online]. Available: <https://doi.org/10.1145/3763076>
- [42] J. Ko, Y. Kook, and K. Shin, “Incremental lossless graph summarization,” in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, p. 317–327. [Online]. Available: <https://doi.org/10.1145/3394486.3403074>
- [43] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. M. Bronstein, “Temporal graph networks for deep learning on dynamic graphs,” *CoRR*, vol. abs/2006.10637, 2020. [Online]. Available: <https://arxiv.org/abs/2006.10637>
- [44] Paradigm, “Reth,” <https://github.com/paradigmxyz/reth>, 2025.