Unveiling Ethereum Mixing Services Using Enhanced Graph Structure Learning

Yan Wu, *Member, IEEE*, Cong Wu, *Member, IEEE*, Yebo Feng, Lin Li, Xinyue Zhang, Zhen Li, Jiahang Sun, *Graduate Student Member, IEEE*, Zijian Zhang, *Senior Member, IEEE*, Jincheng An, Yong Liu, Zhitao Guan, *Member, IEEE*, and Liehuang Zhu, *Senior Member, IEEE*

***Abstract*—As cryptocurrency prices continue to recover, crypto crimes such as money laundering are becoming increasingly rampant. Mixing services such as Tornado Cash have become the primary tools for obfuscating illegal financial transactions due to their inherent anonymity mechanisms. Tornado Cash is a non-custodial, smart contract-based mixing service (SC-CMS) that breaks the direct mapping between deposit and withdrawal accounts, hindering regulators from tracking illicit fund flows. Existing deanonymization methods for Tornado Cash suffer from several challenges, including vague theoretical concepts, evolving mixing mechanisms, and insufficient labeled samples. To address these concerns, this paper proposes the first formal concept of SC-CMS to facilitate and evaluate the deanonymization efforts systematically. We design a novel linkability attack, LASC, based on enhanced graph structure learning, to associate mixing accounts on Tornado Cash and mathematically prove its feasibility. Comprehensive experiments on real Ethereum transactions demonstrate that LASC outperforms state-of-the-art works in both performance and efficiency.**

***Index Terms*—Mixing service, address correlation, Tornado cash, graph representation learning, Ethereum.**

I. Introduction

CRYPTOCURRENCY crimes have escalated significantly, with more than 816 hacking incidents between 2022 and 2024, resulting in $7.7 billion in stolen funds [1], [2], [3]. Major exploits targeted platforms such as Ronin Network [4], Wormhole [5], Horizon Bridge [6], Beanstalk Farm [7], and Nomad [8]. The Ronin Network and Wormhole attacks are among the seven largest cryptocurrency thefts to date, involving hundreds of millions of dollars [9]. A key laundering method involves converting stolen tokens to Ether and routing them through Tornado Cash [10], [11], a non-custodial mixing service. Tornado Cash uses cryptographic techniques to sever on-chain links between deposit and withdrawal transactions, obscuring fund flows. This mechanism significantly hinders law enforcement's ability to track stolen funds and disrupts regulatory oversight.

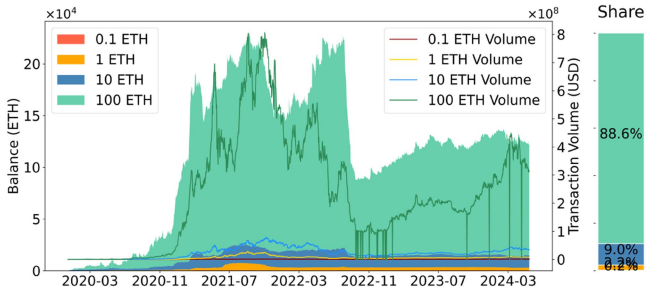


Fig. 1. Balances and Transaction Volume of Tornado Cash over Time.

Conventional financial regulations, such as Know Your Customer (KYC) guidelines, are ineffective against crypto crimes facilitated by Tornado Cash due to its anonymity mechanisms [12], [13]. KYC requires institutions to verify customer identities to mitigate risks such as money laundering and terrorism financing [14]. Blockchain technology, however, enables decentralized transactions that do not require users to disclose their real identities [15], [16], [17]. Tornado Cash operates outside KYC regulations, leveraging the legal ambiguity surrounding emerging technologies to enhance user privacy. As depicted in Fig. 1, Tornado Cash has processed over 511,000 ETH (approximately $1.79 billion) since its launch in 2019, much of it linked to illicit activities [18], [19], [20]. Hackers obfuscate fund flows through Tornado Cash to bypass regulatory oversight, complicating efforts to trace stolen assets [21], [22].

Tornado Cash enables cryptocurrency mixing through smart contracts, where each contract allows users to deposit and withdraw only one fixed denomination of a specific cryptocurrency.

Received 29 March 2025; revised 4 November 2025; accepted 26 November 2025. Date of publication 8 December 2025; date of current version 12 March 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62232002, in part by the Joint Key Project of the National Natural Science Foundation of China under Grant U2468205, and in part by the National Natural Science Foundation of China under Grant 62372173. *(Corresponding authors: Zijian Zhang; Yong Liu.)*

Yan Wu, Xinyue Zhang, Zijian Zhang, and Liehuang Zhu are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: wuyan.bit@gmail.com; xinyuez398@gmail.com; zhangzijian@bit.edu.cn; liehuangz@bit.edu.cn).

Cong Wu is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China (e-mail: cnacwu@whu.edu.cn).

Yebo Feng is with the College of Computing and Data Science, Nanyang Technological University, Singapore 639798 (e-mail: yebo.feng@ntu.edu.sg).

Lin Li is with National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100094, China (e-mail: lilin@cert.org.cn).

Zhen Li and Jiahang Sun are with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: zhen.li@bit.edu.cn; sunjh@bit.edu.cn).

Jincheng An is with QAX Security Center, Qi-AnXin Technology Group Inc., Beijing 100000, China (e-mail: anjincheng@qianxin.com).

Yong Liu is with Zhongguancun Laboratory, Qi An Xin Technology Group Inc., Beijing 100000, China (e-mail: liuyong03@qianxin.com).

Zhitao Guan is with the School of Control and Computer Engineering, North China Electric Power University, Beijing 100000, China (e-mail: guan@ncepu.edu.cn).

Digital Object Identifier 10.1109/TDSC.2025.3640941

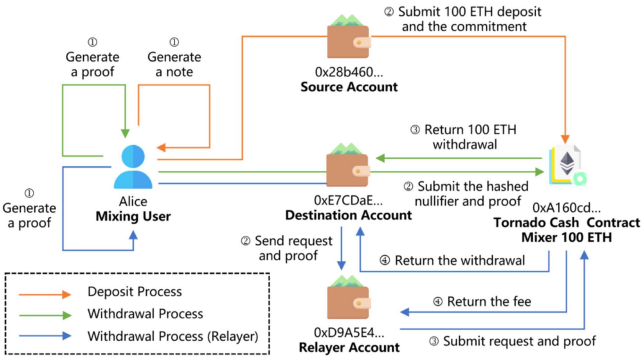


Fig. 2. User Interaction Process of Tornado Cash.

As shown in the orange line in Fig. 2, when depositing, the user submits the hash of a locally generated note (i.e., commitment) along with the funds to the mixing contract from their account [23]. The deposit note must remain confidential, as it serves as a private key to the deposited funds and can only be redeemed in full at one time without any splitting into parts. When withdrawing, the user generates a zero-knowledge proof using the nullifier and secret in the note to ensure that the withdrawal does not reveal the deposit's origin. After validating that the note has not been reused via the proof and nullifier, the "clean" funds are transferred in full to the recipient's account. This process achieves unlinkability between deposit and withdrawal transactions.

Tornado Cash uses a Merkle tree to manage its mixing pool, where leaf nodes store hashed commitments of all deposits, and the tree is updated with every deposit or withdrawal [24]. The anonymity of Tornado Cash depends on the size of the anonymity set, as a larger pool of simultaneous users makes it cryptographically infeasible to link deposit and withdrawal commitments. If only one valid commitment exists in the pool, it becomes trivial to associate the source and destination, even with different accounts. To enhance privacy further, Tornado Cash introduces optional services such as *relayer* addresses and contracts like *old proxy*, *proxy*, and *router*. As shown by the blue line in Fig. 2, relayers enable gas-free withdrawals by executing transactions on behalf of users, eliminating the need to pre-deposit fees into withdrawal accounts. The *old proxy* and *proxy* contracts facilitate indirect withdrawal interactions between users with mixing contracts, while the *router* contract extends these functions to support deposits and withdrawals, making tracing illicit funds more complicated.

While plenty of deanonymization methods [25], [26], [27], [28], [29] for Tornado Cash have been suggested to address its misuse in illegal money laundering activities, these approaches still encounter several significant challenges.

- Existing approaches lack a unified and formal definition of smart contract-based mixing services (SC-CMS), the underlying technology of Tornado Cash, where they primarily concentrate on the implementation details of Tornado Cash's deanonymization techniques. The diverse interpretations and descriptions of these services hamper the systematic evaluation of the relevant works. The ambiguous security requirements limit theoretical exploration on the feasibility of deanonymization technology.
- Existing work ignores the detrimental effects of deanonymization arising from the evolving nature of Tornado Cash. The mixing mechanism is constantly updated, incorporating new technologies and strategies, such as *old proxy*, *proxy*, and *router* auxiliary contracts, to enhance user privacy. The original mixing implementation for simple direct deposits and withdrawals has shifted to complex intermittent transactions through puppet accounts. These changes render the characteristics of the original deanonymization method no longer applicable, weakening the ability to identify mixing accounts and their relationships on Tornado Cash.
- Existing methods struggle with the scarcity of reliable samples, limiting their generalization to real-world scenarios. Since blockchain address associations are rarely public, especially when mixing users prioritize anonymity, available research data is even more scarce. For the recent research, heuristic-based methods lack verification of the ground truth set, and neural network-based approaches rely on small-capacity sample sets (approximately 110 pairs), which reduces the effectiveness of their deanonymization results.

To bridge these gaps, this paper focuses on the deanonymization of SC-CMS represented by Tornado Cash, attempting to combat illicit activities by associating mixing accounts. Specifically, we formalize the concept of SC-CMS, the underlying technology of Tornado Cash, and present a security model that incorporates the desirable properties of unforgeable deposit notes and unlinkable mixing accounts (mainly against malicious third parties). This formal concept allows us to view the services systematically for easier reasoning about their deanonymization approaches. We design a linkability attack named LASC to associate mixing accounts on Tornado Cash, which is proven to be feasible both theoretically and practically. In this attack scenario, six patterns of mixing implementation are first derived from the mixing mechanism of Tornado Cash to restore the real mixing accounts and transfer paths. A mixing transfer graph (MTG) is constructed to represent the mixing and non-mixing interactions between mixing accounts and their neighbors. The heterogeneous graph not only contains the original information of mixing transactions, like the quantity, time, and amount, but also extracts additional important features of patterns, non-mixing transactions with neighbors, and the topological structure. Then, the correlation between mixing accounts can be inferred using enhanced graph representation learning models combined with a richer ground-truth dataset from the Ethereum name service (ENS). Comprehensive experiments demonstrate the superiority of the proposed attack regarding performance.

Our contributions are summarized below:

- The notion of SC-CMS is formally defined, including its syntax, security model, and goals. It enables an abstract investigation of such services and provides a theoretical basis for the feasibility of deanonymization technology.
- A novel linkability attack named LASC is proposed to achieve mixing account correlation on Tornado Cash,

TABLE I
COMPARISON OF MIXING ACCOUNT CORRELATION METHODS ON TORNADO CASH

	[25]	[26]	[27]	[28]	[36]	[29]	Ours
MMF*	○	○	○	◐	○	○	●
MIM*	○	○	○	○	○	◐	●
GTC*	○	○	◐	○	◐	◐	●
SEA*	○	○	○	○	◐	◐	●

* MMF: Mixing Mechanism Formalization, MIM: Mixing Implementation Modeling, GTC: Ground Truth Construction, SEA: Structural Enhancement Analysis.
* ○: dissatisfaction, ◐: partial satisfaction, ●: satisfaction.

featuring three modules: mixing data preparation, dominated by transfer path restoration; heterogeneous mixing transfer graph construction; and mixing account correlation based on enhanced graph structure learning.
- A robust ground-truth dataset for Tornado Cash deanonymization is constructed, which expands the number of reliable account pairs by nearly an order of magnitude.
- A comprehensive evaluation of LASC on real Ethereum transactions is conducted, showing that the proposed method outperforms state-of-the-art methods in terms of performance and efficiency.

II. RELATED WORK

The misuse of cryptocurrency mixing services in illicit online activities has attracted widespread attention, prompting a deep exploration of mixing address association methods. Early research [30], [31], [32] predominantly focused on Bitcoin mixing services, such as Bitcoin Fog, BitLaundry, and Helix. These services operate centrally, with an untrusted third party performing the mixing process. Techniques involving taint analysis, social engineering, and machine learning have been widely utilized to link mixing user addresses and track dubious financial flows [33], [34], [35]. However, differences in the data model and operational mechanisms between Bitcoin and Ethereum render these deanonymization techniques unsuitable for Tornado Cash, an Ethereum mixing service. The decentralized and non-custodial nature of Tornado Cash introduces unique challenges that call for innovative strategies to achieve effective deanonymization. Table I briefly reviews the research on Tornado Cash deanonymization.

Beres et al. [25] first studied the address correlation of Ethereum mixing transactions in the account-based data model. Three heuristic rules are devised to match Tornado Cash's deposit and withdrawal transactions through address reuse, unique gas values, and external transaction links. The authors fully implemented the first two rules but earmarked the generalization of the third rule for future research. Their paper also introduces a quasi-identifier-based user deanonymization method for Ethereum. It leverages a node embedding algorithm in conjunction with multiple quasi-identifiers, such as time-of-day activity, transaction fees, and transaction graph patterns, to conduct user profiling. Tang et al. [26] observed that Ethereum transactions with the same unique gas value are often initiated by the same address. It suggests that the gas price-based rule [25] may have limited practical significance, as its deanonymization results are almost similar to those of the address reuse-based rule. They conducted a statistical analysis of Tornado Cash transactions with different mixing amounts and proposed three heuristic rules derived from novel transaction patterns (i.e., single deposit & withdrawal, multiple pairs of single deposit & withdrawal, multiple deposits & withdrawals). Youn et al. [27] have investigated the use of Tornado Cash for money laundering. They cluster deposit accounts, infer potential criminal accounts through heuristics, and calculate the percentage of deposit funds associated with these addresses. An empirical analysis of many illegal incidents was conducted to identify the types and characteristics of cyberattacks that launder money through the platform. Wang et al. [28] conducted an empirical analysis of two popular zero-knowledge proof-based mixing services, Tornado Cash and Typhoon, investigating their use on multiple blockchains and the impact of OFAC sanctions and anonymity mining on them. The authors use five heuristics similar to the methods in [25], [26] to more accurately measure the size of the anonymity set. Heuristic-based methods have not specifically analyzed the mixing mechanism and implementation of Tornado Cash and only use the regular transaction characteristics of deposit and withdrawal behavior for the association. They might be effective in scenarios where user carelessness leads to poor anonymity but can falter when the target entities and operations change. The absence of valid, real-world datasets further complicates the performance evaluation of these methods.

Beyond heuristic methods, several studies have used neural networks to perform a deanonymization analysis on Tornado Cash. Hu et al. [36] developed a pre-trained transformer for universal fraud detection on Ethereum. The deanonymization of Tornado Cash served as a testbed for evaluating the model's efficacy, which outperforms other DeepWalk-based and graph neural network (GNN)-based methods, such as Diff2Vec and GraphSAGE. Diverging from the time sequence-focused approach of Hu et al. [36], the method [29] aims to deanonymize through the topological structure in a mixing transaction graph. The account correlation task of Tornado Cash is reformulated as a link prediction problem within the constructed graph, which employs a GNN-based prediction model to solve. Unfortunately, these neural network-based methods only capture the structural features of the sequence and the neighborhood among transactions to perform a deanonymization analysis. They are confined by the long-range dependency modeling of the Transformer or the node-centric message-passing paradigm of GNN, which ignores the pairwise structure of the target nodes. These methods strive to collect real data from ENS, but the scarce sample data harms the model's effectiveness. In addition, Du et al. [29] briefly discussed the application of relayers and proxies but did not thoroughly analyze the mixing implementation, especially regarding privacy enhancement techniques such as *old proxy*, *proxy*, and *router* contracts.

Table I comprehensively compares LASC with prior work. Unlike existing studies limited to technical implementation, we

establish the first formal framework for SC-CMS via cryptographic syntax and security games, resolving the theoretical feasibility gap in mixing service deanonymization. While Du et al. [29] observed relayers and proxies but failed to mitigate their privacy impact, LASC decodes six mixing patterns to eliminate noise and recover true mixing accounts. Moreover, we release the first public dataset to solve critical data scarcity, which is over nine times larger than [29]. Both schemes in [29], [36] utilize graph learning to link accounts, which suffer from GNN's node-centric bias and the transformer's neglect of local contexts. Despite adopting a similar methodology to theirs, LASC overcomes the above limitations to achieve superior performance by integrating a double radius node labeling (DRNL)-enhanced graph convolutional network (GCN) and a binary structure transformer with mixing implementation patterns and non-mixing transaction features.

III. Preliminary

This section briefly introduces the fundamental knowledge of Tornado Cash and the Ethereum Name Service.

A. Tornado Cash

Tornado Cash implements cryptocurrency mixing using cryptographic techniques and smart contracts [23]. Its smart contracts are categorized into two types: mixing contracts and auxiliary contracts. Mixing contracts handle core deposit and withdrawal functions that allow users to deposit a fixed denomination (e.g., 0.1 ETH, 1 ETH, 10 ETH, 100 ETH) into a fund pool and withdraw an equal amount of "clean" coins, severing the on-chain link between source and destination accounts. Auxiliary contracts, such as *old proxy*, *proxy*, *router*, and *relayer registry*, support indirect interactions between users and mixing contracts to further obfuscate transaction paths. Unlike proxy contracts that users trade directly, the *relayer registry* contract provides a list of active registered relayer accounts from which users can select a relayer to complete the withdrawal process on their behalf. These mechanisms collectively strengthen the ability to obfuscate fund flows and resist transaction tracing on Tornado Cash.

In addition to smart contracts, cryptographic techniques such as zero-knowledge proofs, hash functions, and Merkle trees underpin Tornado Cash's mixing implementation. The Groth16 algorithm [37], a typical zero-knowledge succinct non-interactive argument of knowledge (zk-SNARKs) scheme, is instrumental in maintaining the unlinkability between deposit and withdrawal transactions. It enables a user with an available commitment to compute a hard-to-forge proof on an elliptic curve point derived in a trusted setting, while the contract can quickly check the proof to withdraw funds. The proof represents the user's ownership of the deposit asset without revealing any information about the note commitment.

Tornado Cash leverages two cryptographic hash functions, Pedersen [38] and MiMC [39], for secure commitments and efficient Merkle tree hashing. The Pedersen hashed commitment works as a secret key to the deposit, easy to verify but hard to reverse back into the original note. Its calculation lies on the Baby Jubjub elliptic curve [40], which is of the same order as the BN128 elliptic curve [41], [42]. The operation enjoys gas efficiency due to the precompile support of Ethereum for these curves [43]. The MiMC hash function is used to compute the leaf node labels of a Merkle tree. Its non-parallelizable property ensures that the MiMC hashed result is difficult to compute but easy to verify. MiMC has low multiplication complexity over prime fields, contributing to gas efficiency. These hash functions improve the security of the mixing implementation by making it computationally infeasible to forge commitments with collision paths in the Merkle tree.

Each Tornado Cash mixing contract maintains a specialized Merkle tree [44] with 20 levels to track valid commitments associated with that contract. The leaf nodes of the tree are labeled with MiMC-hashed commitments, allowing efficient verification without exposing the content. At the initialization of the tree, a unique path is pre-allocated beyond the tree height, starting from a "zero leaf" node with a distinctive label. Each subsequent non-leaf node up to the root is labeled as if the entire bottom of the tree were filled with identical leaf nodes. With each new deposit, the contract integrates the new commitment into the Merkle tree, updating the tree's index and generating a new "root" that reflects the latest state of the tree. The updated index determines whether the next commitment is inserted to the left or right of its Merkle path entry. The new "root" is added to a rolling history containing the labels of the last 100 root nodes, which is used in the zk-SNARK proof to verify withdrawals.

B. Ethereum Name Service

Ethereum Name Service (ENS) is an essential infrastructure within the Ethereum ecosystem that serves as a decentralized blockchain-based naming system [45]. Drawing inspiration from the conventional Domain Name System (DNS), it provides a mapping between human-readable identifiers, such as domain names, and Ethereum addresses or other resource identifiers on the blockchain.

Within the ENS framework, three pivotal roles are defined: (1) *Registrant* (i.e. *owner*) is the account that initially registers the ENS domain, responsible for establishing and maintaining control over the domain name. (2) *Controller* (i.e. *manager*) is the account that wields the authority to manage the ENS domain, responsible for updating the records associated with the domain name. (3) *Resolver* is a smart contract integral to the ENS system, responsible for resolving ENS names into addresses or other data on the blockchain.

The workflow of ENS typically proceeds as follows. A user begins by registering an ENS domain and then becomes the registrant of that domain. The registrant then has the prerogative to appoint a controller, who may be the same as the registrant or a distinct account. Armed with the ability to configure the resolver, the controller determines the mapping process of the ENS name to its corresponding on-chain or off-chain data.

IV. Formal Definitions of SC-CMS

This section defines the notion of SC-CMS, including syntax, security, and goals. For ease of understanding, we assume

TABLE II
THE NOTATIONS

Notations	Descriptions
tx, tx'	the Ethereum external/internal transaction
hx	the transaction hash (a unique 66-character identifier)
ts	the transaction timestamp
from	the spending party of the transaction (creator)
to	the receiving party of the transaction (recipient)
tv	the value being transacted in Ether
gl	maximum amount of gas allocated for the transaction
gp	the price per unit of gas on date of transaction
gu	the amount of gas eventually used
input	additional data included for a transaction
$\mathcal{T}^*$, $\mathcal{T}$, $\mathcal{T}'$	the set of full/external/internal transactions
tx^d, tx^w	the deposit/withdrawal transaction of Tornaodo Cash
cmt	the commitment of a deposit note
pf	the zk-SNARK proof of a withdrawal request
rt	the root of the Merkle tree
nh	the hash value of nullifier
rl	the relayer account delegated to interact with the contract
rfee	the amount paid to the relayer
rfund	the amount refunded to a recipient
addr	the blockchain address of an Ethereum account
curr	the currency of mixing fund
amt	the fixed mixing amount of a Tornado Cash mixing contract
nt	the deposit note stored privately

that the SC-CMS scheme's contract is deployed on Ethereum. Ethereum transactions with their sets are formulated as in Definitions 1-3. Table II lists common notations.

A. Syntax

An SC-CMS scheme allows users to deposit and withdraw funds by transacting with a mixing service deployed on the blockchain as a smart contract, thereby enhancing the anonymity of cryptocurrency fund transfers. It is generally composed of four probabilistic polynomial-time (PPT) algorithms: **SerInit**, **Deposit**, **Withdrawal**, and **UnlinkJudge**.

- $\mathcal{S} \leftarrow \mathbf{SerInit}(1^\lambda)$: Taking the security parameter 1^λ as input, the service initialization algorithm **SerInit** outputs the mixing service's smart contract $\mathcal{S}$ with deposit and withdrawal functions that work properly on the blockchain.
- $(tx^d, nt) \leftarrow \mathbf{Deposit}(\mathcal{S}, curr, amt, addr^d)$: Taking the deposit information (i.e., the contract $\mathcal{S}$, a currency *curr*, a fixed amount *amt*, and source address $addr^d$) as input, the deposit algorithm **Deposit** outputs a deposit transaction tx^d publicly recorded on the blockchain and a deposit note *nt* privately stored by the user. The note can be treated as the private key to this deposit asset.
- $\{tx^w, \perp\} \leftarrow \mathbf{Withdrawal}(\mathcal{S}, nt, curr, amt, addr^w)$: Taking a note *nt* and the corresponding withdrawal information (i.e., the contract $\mathcal{S}$, the currency *curr*, the fixed amount *amt*, and destination address $addr^w$) as input, the withdrawal algorithm **Withdrawal** outputs a withdrawal transaction tx^w published on the blockchain if the note is available or aborts $\perp$ otherwise.
- $b := \mathbf{UnlinkJudge}(tx_i^d, tx_j^w)$: Taking a deposit and withdrawal transaction pair (tx_i^d, tx_j^w) as input, the unlinkability judgment algorithm **UnlinkJudge** outputs a bit b, with $b = 1$ meaning *unlinked* (i.e., $nt_i \neq nt_j$, tx_j^w is not generated by the note corresponding to tx_i^d), and $b = 0$ meaning *linked* (i.e., $nt_i = nt_j$).

The existing literature [34] generally describes the implementation of cryptocurrency mixing services as a three-step process: deposit, mixing, and withdrawal. The introduction of smart contracts binds the deposit and mixing steps, improving the efficiency and security of the mixing process. This paper treats the mixing operation as a natural extension of the deposit operation to reflect the execution characteristics of the SC-CMS more accurately. Moreover, to formally define the security model of the mixing service, we set the unlinkability judgment as an inherent algorithm of SC-CMS. The desired output of this algorithm is always 1, which means that given a pair of Tornado Cash deposit and withdrawal transactions, any external analyst can hardly determine their association, even if they are linked. To achieve the unlinkability goal, the mixing service ensures that the deposit account secretly stores the note used as the judgment condition, and the withdrawal account that shares the note withdraws the mixing fund without revealing the explicit information of the previous deposit transaction.

Definition 1 (Ethereum External Transaction): An Ethereum external transaction refers to a transaction initiated by an externally owned account on the Ethereum platform. It can be expressed as a nine-tuple $tx = (hx, ts, from, to, tv, gl, gp, gu, input)$ and the external transaction set is expressed as $\mathcal{T}$.

Definition 2 (Ethereum Internal Transaction): An Ethereum internal transaction refers to a transaction that occurs within a smart contract on Ethereum, which involves transferring Ether between contracts, invoking other contracts, or executing complex contract operations. It can be expressed as a four-tuple $tx' = (hx, from, to, tv)$ and the internal transaction set is expressed as $\mathcal{T}'$. All internal transactions are triggered by the execution of contract code, usually as a response to an external transaction. An external transaction can result in zero or more internal transactions. If an external transaction *tx* triggers two internal transactions tx'_1 and tx'_2, then: (1) the transaction hash of the two internal transactions is the same as that of the external transaction, i.e., $tx'_1.hx = tx.hx$, $tx'_2.hx = tx.hx$; (2) the total value of the two internal transactions equals the value of the external transaction, i.e., $tx.tv = tx'_1.tv + tx'_2.tv$.

Definition 3 (Ethereum Full Transaction Set): The Ethereum full transaction set refers to the union of the Ethereum external transaction set and the corresponding Ethereum internal transaction set. It can be expressed as $\mathcal{T}^* = \mathcal{T} \cup \mathcal{T}'$, where: (1) $\forall tx'_j \in \mathcal{T}'$, $\exists tx_i \in \mathcal{T}$, s.t. $tx'_j.hx = tx_i.hx$; (2) $\forall tx_i \in \mathcal{T}$, $\sum_{tx'_j \in \mathcal{T}', tx'_j.hx = tx_i.hx} tx'_j.tv = tx_i.tv$.

B. Security Model

1) Correctness: The requirement for the correctness of withdrawal is formulated in Definition 4.

Definition 4: If **Withdrawal**(**Deposit**($\mathcal{S}$, *curr*, *amt*, $addr^d$), *curr*, *amt*, $addr^w$, $\mathcal{S}$) $= tx^w$ holds for all security parameters $\lambda \in \mathbb{N}$, then an SC-CMS scheme is perfectly correct for λ.

2) Security Model: To complete the cryptocurrency mixing service and achieve the desired anonymity, the SC-CMS scheme needs to satisfy two security properties: unforgeability and unlinkability of cryptocurrency fund transfers. For unforgeability, there exists an adversary $\mathcal{A}_1$ who has access to multiple

legitimate deposit and withdrawal transactions of the mixing service and tries to forge a valid note to withdraw mixing funds without depositing any assets. For unlinkability, the SC-CMS scheme is vulnerable to linkability attacks from an adversary $\mathcal{A}_2$, who accesses all transactions on the blockchain and interacts with the mixing service. We formalize the above properties through two interactive games, where the interaction between an adversary and multiple oracles represents the adversary's capability to interact with the SC-CMS scheme.

Game I: This game discusses $\mathcal{A}_1$ and the unforgeability security of cryptocurrency fund transfers for SC-CMS schemes.

Initialization Phase: The challenger $\mathcal{C}$ publishes $\mathcal{S} \leftarrow \mathbf{SerInit}(\lambda)$ to $\mathcal{A}_1$.

Query Phase: In this phase, $\mathcal{A}_1$ can adaptively query the following oracles:

- *Deposit* $\mathcal{O}_{\textit{Deposit}}$*:* Given $(\mathcal{S}, curr, amt, addr^d)$, $\mathcal{C}$ returns $(nt, tx^d) \leftarrow \mathbf{Deposit}(\mathcal{S}, curr, amt, addr^d)$ to $\mathcal{A}_1$. Let $\mathcal{T}_q^d$ denote the transaction set that $\mathcal{A}_1$ has queried the oracle.
- *Withdrawal* $\mathcal{O}_{\textit{Withdrawal}}$*:* Given $(\mathcal{S}, nt, curr, amt, addr^w)$, $\mathcal{C}$ returns $tx^w \leftarrow \mathbf{Withdrawal}(\mathcal{S}, nt, curr, amt, addr^w)$ to $\mathcal{A}_1$. Let $\mathcal{T}_q^w$ denote the transaction set that $\mathcal{A}_1$ has queried the oracle.
- *UnlinkJudge* $\mathcal{O}_{\textit{UnlinkJudge}}$*:* Given (tx_i^d, tx_j^w), $\mathcal{C}$ returns 0 if tx_i^d and tx_j^w are linked, or 1 otherwise.

Forgery Phase: $\mathcal{A}_1$ outputs a forgery (tx^{d*}, nt^*) after all queries. $\mathcal{A}_1$ is considered to succeed if and only if (a) tx^{d*} has never been obtained by queries; (b) **Withdrawal**$(\mathcal{S}, nt^*, curr, amt, addr^w) \rightarrow tx^w$ holds.

We say that an SC-CMS scheme satisfies the unforgeability of security properties if no efficient adversary can succeed in the above experiment with non-negligible probability:

Definition 5 (Unforgeability of Cryptocurrency Fund Transfers): An SC-CMS scheme Π = (**SerInit**, **Deposit**, **Withdrawal**, **UnlinkJudge**) is secure under existential unforgeability against adaptive chosen-deposit attacks if there exists a negligible function negl for all PPT adversaries $\mathcal{A}_1$ such that:

$$\begin{aligned} &Adv^{\mathsf{Unforge}}_{\mathcal{A}_1,\mathsf{SC\text{-}CMS}} \\ &= |\mathsf{Pr}[\mathbf{Withdrawal}(\mathcal{S}, nt^*, curr, amt, addr^w) \rightarrow tx^w]| \\ &\leq \mathsf{negl}(\lambda) \end{aligned}$$

Game II: This game discusses $\mathcal{A}_2$ and the unlinkability security of cryptocurrency fund transfers for SC-CMS schemes.

Initialization Phase: $\mathcal{C}$ publishes $\mathcal{S} \leftarrow$ **SerInit**(λ) to $\mathcal{A}_2$.

Query Phase: In this phase, $\mathcal{A}_2$ adaptively queries $\mathcal{O}_{\mathsf{Deposit}}$, $\mathcal{O}_{\mathsf{Withdrawal}}$, and $\mathcal{O}_{\mathsf{UnlinkJudge}}$ similar to **Game I**.

Challenge Phase: $\mathcal{A}_2$ chooses (*curr*, *amt*). $\mathcal{C}$ replies with the unqueried deposit and withdrawal transaction sets $\mathcal{T}_c^d, \mathcal{T}_c^w$ of the specified (*curr*, *amt*), where $|\mathcal{T}_c^d| = m \geq n = |\mathcal{T}_c^w|, m, n \in \mathbb{N}_+$.

Output Phase: $\mathcal{A}_2$ outputs a guess (tx_g^d, tx_g^w). $\mathcal{A}_2$ is considered to succeed if and only if: (1) $tx_g^d \in \mathcal{T}_c^d, tx_g^w \in \mathcal{T}_c^w, |\mathcal{T}_c^d| = m \geq n = |\mathcal{T}_c^w|$, $m, n \in \mathbb{N}_+$; (2) **UnlinkJudge**$(tx_g^d, tx_g^w) = 0$ hold.

We say that an SC-CMS scheme satisfies the unlinkability security property if no efficient adversary can succeed in the above experiment with a non-negligible probability:

Definition 6 (Unlinkability of Cryptocurrency Fund Transfers): An SC-CMS scheme Π = (**SerInit**, **Deposit**, **Withdrawal**, **UnlinkJudge**) is unlinkability secure if there exists a negligible function negl for all PPT adversaries $\mathcal{A}_2$ such that:

$$\begin{aligned} Adv^{\mathsf{Unlink}}_{\mathcal{A}_2,\mathsf{SC\text{-}CMS}} &= |\mathsf{Pr}[\mathbf{UnlinkJudge}(tx_g^d, tx_g^w) = 0]| \\ &= |\frac{1}{m}| \leq \mathsf{negl}(m) \end{aligned}$$

Definition 7: An SC-CMS scheme is computationally secure for the security parameter λ if it is correct for λ as in Definition IV-A, and any PPT adversary wins the games of security requirements only with a negligible advantage.

C. Goals

To provide users with satisfactory privacy-enhancing mixing functions for cryptocurrency transactions, an SC-CMS scheme should achieve a series of system goals.

- *Authentication:* Mixing funds must only be withdrawn by users with the corresponding note and cannot be spent more than once during the mixing process. An SC-CMS scheme achieves this goal if it is EUF-CMA secure as described in Definition 5.
- *Anonymity:* The on-chain link between the creator and the recipient of a cryptocurrency transaction must be decoupled, making the sources and destinations of funds hard to trace. An SC-CMS scheme achieves this goal if it is unlinkability secure, as described in Definition 6.
- *Efficiency:* The execution operations of a mixing service should have low latency and transaction fees, making it efficient and economically feasible for users to mix coins.
- *Decentralization:* The mixing process is governed by smart contracts to ensure that each operation is executed transparently and accurately, reducing the risks of single-point failure and unequal treatment.
- *Scalability* The service can handle numerous transactions simultaneously and adapt to complex network conditions without compromising performance or security.
- *Usability:* The service should have a user-friendly interface and operation while being compatible with various wallets and cryptocurrencies, particularly within the ecosystem that smart contracts build on.

V. Problem Statement

This section describes the construction of Tornado Cash and proposes a linkability attack, LASC, that can theoretically break its anonymity property.

A. The Construction of Tornado Cash

SerInit: Given the security parameter λ, the development team deploys the mixing contracts on the Ethereum platform to complete the initialization settings.

- Select an appropriate elliptic curve $\mathbb{F}_q$ over a large prime number q, and three groups $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ of prime order q with a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.
- Pick H_p, H_m as the Pedersen and MiMC hash functions.
- Utilize the multi-party computation ceremony to produce $(\sigma, \tau) \leftarrow$ **Setup**(λ, C) for each withdrawal.
- Construct a specific Merkle tree with initialization settings as described in Section III-A. Let $\mathcal{L}_{nt}, \mathcal{L}_{rt}$ denote the used note list and the generated root list, respectively.

Deposit: To complete a deposit operation, a user interacts with a mixing contract $\mathcal{S}_{curr,amt}$, which mixes a fixed amount *amt* of a specific currency *curr* for each user at a time.

- Choose $n, s \in \mathbb{Z}_q^*$ at random and keep $nt = (n, s)$ a secret.
- Call the deposit function of $\mathcal{S}_{curr,amt}$ with *cmt* as input through $addr_d$, where $cmt = H_p(nt)$.
- $tx^d, \mathcal{S}_{curr,amt}$ inserts *cmt* into the Merkle tree as a new leaf node with its label $H_m(cmt)$ and updates the recalculated root to $\mathcal{L}_{rt}$.

Withdrawal: To complete a withdrawal operation, a user interacts with a mixing contract $\mathcal{S}_{curr,amt}$.

- Derive $\mathcal{L}_{pe}$ and $\mathcal{L}_{pi}$ from *nt*, where $\mathcal{L}_{pe}$ and $\mathcal{L}_{pi}$ represent the node label set existing in the path between the root node and the leaf node of the Merkle tree, as well as the 0/1 value array indicating whether each specified path element is located on the left or right of its parent node, respectively.
- Utilize public inputs ϕ_p and private inputs ϕ_s to compute the witness ω through three checks and generate a proof of the withdrawal request $\pi \leftarrow$ **Prove**(C, σ, ϕ_p, ω), where $\phi_p = \{rt, nh, to, rl, rfee, rfund\}$ and $\phi_s = \{nt, \mathcal{L}_{pe}, \mathcal{L}_{pi}\}$. The checks involve (1) nullifier hash check: the publicly provided nullifier hash is a component of the original commitment; (2) Merkle tree check: the commitment exists within some path beneath the specified Merkle root; (3) extra withdrawal parameter check: the provided withdrawal parameters cannot be tampered with in subsequent implementations.
- Call the withdrawal function of $\mathcal{S}_{curr,amt}$ via the withdrawal account $addr_w$ with π and ϕ_p as input.
- $\mathcal{S}_{curr,amt}$ completes tx^d if the following conditions are met: (1) $rfee \leq amt$; (2) $nh \notin \mathcal{L}_{nh}$; (3) $rt \in \mathcal{L}_{rt}$; (4) **Verify**(C, σ, ϕ_p, π) $\to 1$. Then it inserts *nh* into $\mathcal{L}_{nh}$. If any of the conditions are not met, the withdrawal operation aborts.

UnlinkJudge: A user observes the transaction details to determine whether tx_i^d and tx_j^w are unlinked. If the user is a mixing user who creates tx_i^d and tx_j^w, then output 0; otherwise, output 1. If the user is an ordinary individual who knows nothing and cannot identify any characteristics, output 1.

Definition 8 (Tornaodo Cash Deposit Transaction): A Tornado Cash deposit transaction refers to a special Ethereum external transaction in which the user account invokes the deposit function of the Tornado Cash smart contract. It can be expressed as tx^d, where $tx^d.input = (cmt)$. *cmt* denotes a commitment to this deposit note, which shows that the user has created a deposit with a fixed amount and has not yet withdrawn that asset.

Definition 9 (Tornado Cash Withdrawal Transaction): A Tornado Cash withdrawal transaction refers to a special Ethereum external transaction in which the user account invokes the withdrawal function of the Tornado Cash smart contract. It can be expressed as tx^w, where $tx^w.input =$ (*pf*, *rt*, *nh*, *to*, *rl*, *rfee*, *rfund*).

B. Linkability Attack

As discussed in III-A and IV-B, except for the security of the Groth16 algorithm, Tornado Cash relies on a sufficiently large anonymity set to maintain its anonymity. Unfortunately, its anonymity is vulnerable to a new link attack, LASC, launched by an active third-party adversary $\mathcal{A}_{LASC}$. In this attack scenario, the adversary can collect all Ethereum transactions and a few reliable link samples, associating mixing accounts through graph representation learning. Specifically, $\mathcal{A}_{LASC}$ leverages the mixing implementation patterns derived from Tornado Cash's mixing mechanism to resist the interference of auxiliary functions, thereby restoring the real mixing accounts and transactions. Then, a mixing account-centric graph is built that contains mixing transactions with mixing contracts and ordinary transactions with neighbor accounts. Thus, the account correlation problem on Tornado Cash is transformed into a link prediction task between mixing account pairs in the graph. To predict the link probability of node pairs as accurately as possible, a well-performing model is trained to learn from the node features and the graph structure.

Theorem 1: The linkability attack LASC can break the anonymity of Tornado Cash if all transactions on Ethereum can be accessed.

Proof: Define a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ based on real mixing accounts with their relevant transactions, where $\mathcal{V}_D, \mathcal{V}_W \subseteq \mathcal{V}$ denote the mixing accounts for deposit or withdrawal. Note that $\mathcal{G}$ is non-random since the mixing transactions are driven by user behaviors that are not uniformly random. This non-randomness introduces structural and temporal biases into $\mathcal{G}$, enabling graph representation learning to capture potential behavioral characteristics. Existing work [25], [26], [29], [36] has validated that such user-induced patterns are indeed detectable and learnable. Given an $addr_w^* \in \mathcal{V}_W$ that initiates $tx_g^w \in \mathcal{T}_c^w/\mathcal{T}_q^w$, the goal of LASC is to infer the correct $addr_d^* \in \mathcal{V}_D$ that initiates $tx_g^d \in \mathcal{T}_c^d/\mathcal{T}_q^d$. Assume that the LASC model is properly trained. For embedding, it learns a mapping $f : \mathcal{V} \to \mathbb{R}^D$ and is optimized to preserve graph topology and node similarities. For inference, it ranks $addr_d \in \mathcal{V}_D$ by similarity $\sigma(f(addr_d), f(addr_w^*))$ and predicts $addr_d' = \arg\max_{addr_d \in \mathcal{V}_D} \sigma(f(addr_d), f(addr_w^*))$.

The model predicts correctly if $\forall addr_d \in \mathcal{V}_D/\{addr_d^*\}$,

$$\sigma(f(addr_d^*), f(addr_w^*)) > \sigma(f(addr_d), f(addr_w^*)). \quad (1)$$

It systematically maps linked nodes closer in the embedding space than random node pairs to create a statistical bias, where true links consistently achieve higher similarity scores than non-links, making correct predictions more frequent than chance. Thus, the success probability of LASC can be decomposed as:

$$Succ_{LASC} = |\mathsf{Pr}[\sigma_{addr_d^*} > \max_{addr_d \neq addr_d^*} \sigma_{addr_d}]|, \quad (2)$$

where $\sigma_{addr_d^*} = \sigma(f(addr_d^*), f(addr_w^*))$ and $\sigma_{addr_d} = \sigma(f(addr_d), f(addr_w^*))$.

Let $D^* = \sigma(f(addr_d^*), f(addr_w^*))$ and $D = \sigma(f(addr_d), f(addr_w^*))$ $(addr_d \neq addr_d^*)$ denote the similarity score distributions of true links and non-links, respectively. The similarity threshold is expressed by t. Since the model optimizes embeddings to maximize similarity for true links, the similarity scores for true links stochastically dominate those for non-links. The stochastic dominance condition

$$\mathsf{Pr}[D^* > t] > \mathsf{Pr}[D > t] \quad \forall t \in [0, 1], \tag{3}$$

holds, which means that the cumulative distribution functions (CDFs) satisfy $F_{D^*}(t) \leq F_D(t)$. Therefore, this relationship directly guarantees:

$$\mathbb{E}[D^*] = \int_0^1 (1 - F_{D^*}(t))\, dt > \int_0^1 (1 - F_D(t))\, dt = \mathbb{E}[D].$$

Then, the order statistics analysis is performed on the distribution of the maximum similarity scores of non-links $D_{\max} = \max\{D_1, \ldots, D_{|\mathcal{V}_D|-1}\}$. Since $D_i (1 \leq i \leq |\mathcal{V}_D| - 1)$ are independent and identically distributed with $F_D(t)$, the CDF of $D_{\max}$ is

$$F_{\max}(t) = \mathsf{Pr}[D_{\max} \leq t] = [F_D(t)]^{|\mathcal{V}_D|-1}. \tag{4}$$

Therefore, the success probability of LASC can become

$$\begin{aligned} Succ_{LASC} &= |\mathsf{Pr}[D^* > D_{\max}]| \\ &= \int_0^1 \mathsf{Pr}[D^* > t]) \cdot f_{\max}(t) dt \\ &> \int_0^1 \mathsf{Pr}[D > t]) \cdot f_{\max}(t) dt \\ &= \frac{1}{|\mathcal{V}_D|}. \end{aligned} \tag{5}$$

where $f_{\max}(t) = \frac{d}{dt} F_{\max}(t)$.

Considering that the success probability of random guessing (RG) is $\frac{1}{|\mathcal{V}_D|}$, the margin between it and that of LASC can be calculated as

$$\begin{aligned} \Delta &= Succ_{LASC} - Succ_{RG} \\ &= Succ_{LASC} - \frac{1}{|\mathcal{V}_D|} \\ &= \int_0^1 (\mathsf{Pr}[D^* > t] - \mathsf{Pr}[D > t]) \cdot f_{\max}(t) dt \\ &= \int_0^1 (F_D(t) - F_{D^*}(t)) \cdot f_{\max}(t) dt. \end{aligned} \tag{6}$$

Δ is non-negligible for the following reasons:

- Δ strictly positive for the properly trained model.
- Δ increases monotonically with the improvement of embedding dimension, training epochs, and graph regularity.

This result demonstrates that Tornado Cash fails to satisfy the unlinkability requirement of Definition 6, as LASC achieves a non-negligible advantage Δ over random guessing $\frac{1}{m}$. Therefore, it is theoretically feasible for LASC to break the anonymity of Tornado Cash.

Algorithm 1: Extracting Direct Deposit Pattern $[\mathcal{P}_a]$.

Input: $\mathcal{T}_{rTCM} \subsetneq \mathcal{T}^*_{rTC}$, where $tx_i = (hx, ts, from, to, tv, gl, gp, gu, input) \in \mathcal{T}_{rTCM}$.
Output: $\mathcal{T}_{sTC\_a}$.

```
initialize T_sTC_a = {}.
for tx_i ∈ T_rTCM do
  if tx_i.tv > 0 and |tx_i.input| = 1 then
    add tx_i in T_sTC_a.
  end if
end for
return T_sTC_a.
```

VI. Methodology

This section describes LASC's deanonymization framework.

A. Overview

As depicted in Fig. 3, the deanonymization strategies of LASC mainly consist of three modules: (1) mixing data preparation: restoring real mixing transactions on Tornado Cash from the Ethereum ledger data and constructing a ground-truth dataset of labeled account pairs; (2) mixing transfer graph construction: converting the mixing data graph (MDG) constructed from extended real mixing transactions into the mixing transfer graph (MTG) through supernode merging and account feature extraction; (3) mixing account correlation: encoding the node features and structural information of the target node pair in the MTG enclosing subgraph and then obtaining the link probability of the targets using the decoder.

B. Mixing Data Preparation

To tackle the data scarcity challenges in deanonymization analysis, we present a new mixing data preparation process that introduces transfer path restoration and label account pairing into data collection and construction.

1) Raw Data Collection: This component collects raw transactions about Tornado Cash and ENS from the Ethereum ledger data. We establish an Ethereum full node to synchronize the entire ledger and parse it into a readable raw transaction dataset $\mathcal{T}^*_{rETH}$ through the Ethereum-ETL tool [46]. This dataset is then matched with the core contracts of Tornado Cash and ENS crawled from Etherscan.io[1] and the service websites[2] to form the corresponding raw transaction datasets $\mathcal{T}^*_{rTC}$ and $\mathcal{T}^*_{rENS}$. The raw transaction dataset of Tornado Cash $\mathcal{T}^*_{rTC}$ consisted of a dataset of mixing contracts $\mathcal{T}^*_{rTCM}$ and a dataset of auxiliary contracts $\mathcal{T}^*_{rTCA}$ involving *old proxy*, *proxy*, and *router* contracts.

2) Transfer Path Restoration: This component cleans up the raw Tornado Cash transactions using behavioral pattern analysis to resist interference from the router, proxies, and relayers, thus restoring the real transfer path for mixing users. Observing the deposit and withdrawal behaviors of the mixing users, we

[1] https://etherscan.io/
[2] https://tornadoeth.cash/, https://ens.domains/

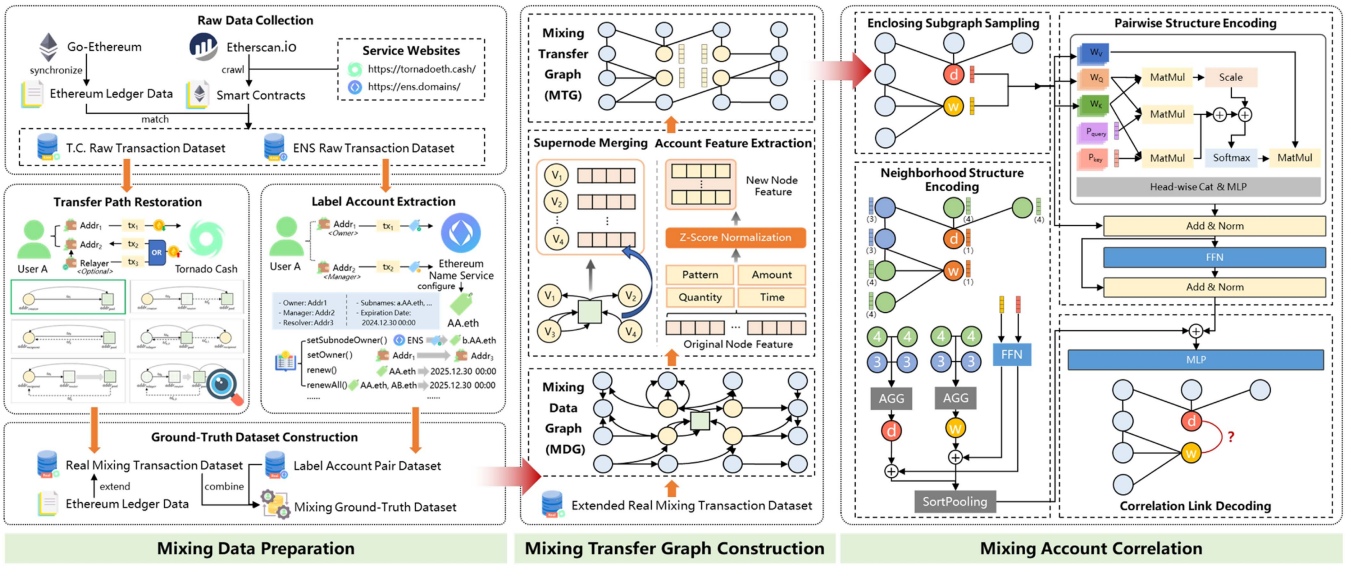


Fig. 3. Deanonymization Strategies of LASC. It mainly involves three modules: mixing data preparation, mixing transfer graph construction, and Mixing Account Correlation.

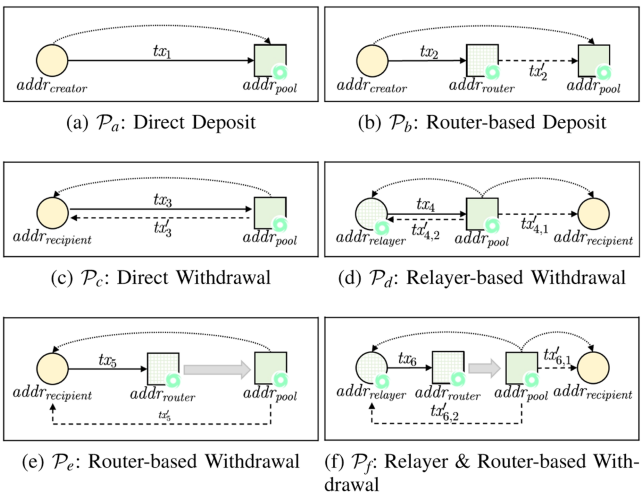


Fig. 4. Usage Patterns of Mixing Implementation on Tornado Cash.

creatively formalize the mixing implementation of Tornado Cash into six usage patterns. Fig. 4 illustrates the interaction between the mixing user and Tornado Cash contracts beneath each usage pattern. From Fig. 4(a) and (b), the deposit operation consists of direct deposit (pattern a) and router-based deposit (pattern b). Pattern a allows funds to be transferred directly from deposit accounts into mixing contracts, so such transactions can be easily extracted from the external transactions of mixing contracts (see Algorithm 1). In contrast, the fund transfer of pattern b is more complicated; that is, users invoke the deposit function of the routing contract to deposit funds into the mixing contract. Its transfer path should be restored by combining the external transactions of the *router* contract with the internal transactions of the mixing contract (Algorithm 2). In addition, there are four patterns for executing withdrawal operations, including direct withdrawal (pattern c), relayer-based withdrawal (pattern d), router-based withdrawal (pattern e), and relayer & router-based withdrawal (pattern f), as illustrated in Fig. 4(c)–(f). Pattern c allows users to call the mixing contract's withdrawal function directly, returning the mixing funds to the designated withdrawal account. For pattern d, the relayer account set by the user initiates a withdrawal transaction, which causes the mixing contract to pay the mixing fund and the rewards to the withdrawal account and the relayer account, respectively. Algorithm 3 selects the above two patterns' transactions from mixing contracts' internal and external transactions. Unlike directly invoking mixing contracts for withdrawal in patterns c and e, the last two patterns interact with the *old proxy*/*proxy*/*router* contract, which triggers mixing contracts to deliver the corresponding coins. The actual withdrawers of these patterns can be identified from the external transactions of the *old proxy*/*proxy*/*router* contract and the internal transactions of the mixing contract through Algorithm 4. All the analysis results of the pattern extraction algorithms are grouped as a dataset of Tornado Cash real mixing transactions $\mathcal{T}_{sTC}$.

3) Label Account Extraction: This component leverages the account correlation heuristics of the key ENS functions to enrich the reliable label account pairs. Existing studies obtain label account clusters from the setOwner and setSubOwner functions of ENS, which appoint a new owner for a domain or subdomain. After revisiting the ENS's core contracts, we find that the controller permissions and domain renewal functions also expose the association between account addresses. An ENS domain's owner (or registrant) can designate a trusted account as a manager (or controller) with the authority to update the ENS record. Due to the trust relationship between these roles, their relevant accounts belong to the same user entity. Furthermore, ownership of ENS domains is time-limited and can usually be obtained through renewal via the renew and renewAll functions. Note that if an account renews a non-expired and non-owned domain, the ownership will not be transferred to it, no matter

Algorithm 2: Extracting Router-based Deposit Pattern [$\mathcal{P}_b$].

Input: $\mathcal{T}_{rTCA}, \mathcal{T}'_{rTCM} \subsetneqq \mathcal{T}^*_{rTC}$, where $tx_i = (hx, ts, from, to, tv, gl, gp, gu, input) \in \mathcal{T}_{rTCA}$ and $tx'_j = (hx, from, to, tv) \in \mathcal{T}'_{rTCM}$.
Output: $\mathcal{T}_{sTC\_b}$.
1: initialize $\mathcal{T}_{sTC\_b} = \{\}$.
2: **for** $tx_i \in \mathcal{T}_{rTCA}$ **do**
3: **if** $tx_i.tv > 0$ **and** $|tx_i.input| = 1$ **then**
4: search tx'_j in $\mathcal{T}'_{rTCM}$ where $tx'_j.hx = tx_i.hx$.
5: add ($tx_i.hx$, $tx_i.ts$, $tx_i.from$, $tx'_j.to$, $tx_i.tv$, $tx_i.gl$, $tx_i.gp$, $tx_i.gu$, $tx_i.input$) in $\mathcal{T}_{sTC\_b}$.
6: **end if**
7: **end for**
8: return $\mathcal{T}_{sTC\_b}$.

Algorithm 3: Extracting Direct Withdrawal Pattern [$\mathcal{P}_c$] and Relayer-based Withdrawal Pattern [$\mathcal{P}_d$].

Input: $\mathcal{T}_{rTCM}, \mathcal{T}'_{rTCM} \subsetneqq \mathcal{T}^*_{rTC}$, where $tx_i = (hx, ts, from, to, tv, gl, gp, gu, input) \in \mathcal{T}_{rTCM}$ and $tx'_j = (hx, from, to, tv) \in \mathcal{T}'_{rTCM}$.
Output: $\mathcal{T}_{sTC\_c}$ and $\mathcal{T}_{sTC\_d}$.
1: initialize $\mathcal{T}_{sTC\_c} = \{\}$ and $\mathcal{T}_{sTC\_d} = \{\}$.
2: **for** $tx_i \in \mathcal{T}_{rTCM}$ **do**
3: **if** $|tx_i.tv| = 0$ **then**
4: **if** $tx_i.input.rr =$ NULL **then**
5: search tx'_j in $\mathcal{T}'_{rTCM}$ where $tx'_j.hx = tx_i.hx$.
6: add ($tx_i.hx$, $tx_i.ts$, $tx'_j.from$, $tx'_j.to$, $tx'_j.tv$, $tx_i.gl$, $tx_i.gp$, $tx_i.gu$, $tx_i.input$) in $\mathcal{T}_{sTC\_c}$.
7: **else**
8: create $\mathcal{T}_t = \{\}$.
9: search $tx'_j \in \mathcal{T}'_{rTCM}$ with $tx'_j.hx = tx_i.hx$ and add the searched tx'_j to $\mathcal{T}_t$.
10: select $tx'_j \in \mathcal{T}_t$ with $tx'_j.to \neq tx_i.input.rr$.
11: add ($tx_i.hx$, $tx_i.ts$, $tx'_j.from$, $tx'_j.to$, $tx'_j.tv$, $tx_i.gl$, $tx_i.gp$, $tx_i.gu$, $tx_i.input$) in $\mathcal{T}_{sTC\_d}$.
12: **end if**
13: **end if**
14: **end for**
15: return $\mathcal{T}_{sTC\_c}$ and $\mathcal{T}_{sTC\_d}$.

Algorithm 4: Extracting Router-based Withdrawal Pattern [$\mathcal{P}_e$] and Relayer & Router-based Withdrawal [$\mathcal{P}_f$].

Input: $\mathcal{T}_{rTCA}, \mathcal{T}'_{rTCM} \subsetneqq \mathcal{T}^*_{rTC}$, where $tx_i = (hx, ts, from, to, tv, gl, gp, gu, input) \in \mathcal{T}_{rTCA}$ and $tx'_j = (hx, from, to, tv) \in \mathcal{T}'_{rTCM}$.
Output: $\mathcal{T}_{sTC\_e}$ and $\mathcal{T}_{sTC\_f}$.
1: initialize $\mathcal{T}_{sTC\_e} = \{\}$ and $\mathcal{T}_{sTC\_f} = \{\}$.
2: **for** $tx_i \in \mathcal{T}_{rTCA}$ **do**
3: **if** $|tx_i.tv| = 0$ **then**
4: **if** $tx_i.input.rr =$ NULL **then**
5: search tx'_j in $\mathcal{T}'_{rTCM}$ where $tx'_j.hx = tx_i.hx$.
6: add ($tx_i.hx$, $tx_i.ts$, $tx'_j.from$, $tx'_j.to$, $tx'_j.tv$, $tx_i.gl$, $tx_i.gp$, $tx_i.gu$, $tx_i.input$) in $\mathcal{T}_{sTC\_e}$.
7: **else**
8: create $\mathcal{T}_t = \{\}$.
9: search $tx'_j \in \mathcal{T}'_{rTCM}$ with $tx'_j.hx = tx_i.hx$ and add the searched tx'_j to $\mathcal{T}_t$.
10: select $tx'_j \in \mathcal{T}_t$ with $tx'_j.to \neq tx_i.input.rr$.
11: add ($tx_i.hx$, $tx_i.ts$, $tx'_j.from$, $tx'_j.to$, $tx'_j.tv$, $tx_i.gl$, $tx_i.gp$, $tx_i.gu$, $tx_i.input$) in $\mathcal{T}_{sTC\_f}$.
12: **end if**
13: **end if**
14: **end for**
15: return $\mathcal{T}_{sTC\_e}$ and $\mathcal{T}_{sTC\_f}$.

how much it pays. Considering the purpose and restrictions of the renewal operation, it can be inferred that the initiator account is associated with the owner account of the renewed domain. Inspired by the above observations, two new heuristic rules for account correlation are designed in Definition 10 and 11. We apply the proposed heuristics and existing rules to ENS raw transactions to derive usable label account pairs $\mathcal{L}_{rENS}$.

Definition 10 (ENS Controller-based Account correlation Heuristic): Given two Ethereum accounts $addr_1$ and $addr_2$, if $addr_1$ is the owner of an ENS domain and designates $addr_2$ as the manager of the same domain, then $addr_1$ and $addr_2$ are linked, i.e., **UnlinkJudge**($addr_1$, $addr_2$) $= 0$.

Definition 11 (ENS Renewal-based Account correlation Heuristic): Given two Ethereum accounts $addr_1$ and $addr_2$, if $addr_1$ invokes ENS's renew or renewAll function to extend the expiration date of one or more ENS domains, where the owner of the domain is $addr_2$, then $addr_1$ and $addr_2$ are linked, i.e., **UnlinkJudge**($addr_1$, $addr_2$) $= 0$.

4) Ground-Truth Dataset Construction: This component matches real mixing transactions and labeled account pairs to construct the desired ground-truth dataset for Tornado Cash deanonymization. The real mixing transaction dataset $\mathcal{T}_{rTC}$ is divided into a deposit transaction subset and a withdrawal transaction subset according to the interaction method. We extract the mixing user accounts from the sub-datasets and select their normal transactions in the Ethereum ledger data as neighbor information for supplementation. The ground-truth mixing dataset $\mathcal{T}_L$ of Tornado Cash is derived by calculating and optimizing the intersection of the real mixing transaction dataset and the labeled account pair dataset.

C. Mixing Transfer Graph Construction

From the usage patterns of mixing users in Section VI-B2, Tornado Cash mixing transactions can essentially be treated as a series of deposit and withdrawal behaviors between mixing users and mixing contracts. This mixing implementation of blurring fund transfer paths complicates the address correlation task of deposit and withdrawal accounts from a single mixing transaction's features. Therefore, we adopt graph representation learning techniques to analyze the complex transactional relationships. Apart from mixing transactions with relatively fixed transaction amounts and frequencies, diversified ordinary transactions between mixing users and their neighbors are also included in the analysis as a comparative supplement. The MTG

is designed to capture the topological structure between the mixing accounts and map the actual path of the fund flow.

Specifically, we first construct the Tornado Cash real transaction dataset obtained in the previous stage into an MDG. It is a heterogeneous graph with various types of information attached to its nodes and links. It is centered on mixing user accounts, which not only call Tornado Cash mixing contracts to access services but also interact with neighbor accounts for ordinary fund transfers. The inherent heterogeneity significantly interferes with the extraction of features related to account association from MDG. In particular, Tornado Cash mixing contracts respond to numerous deposit and withdrawal requests from mixing users as supernodes, which easily leads to the topology of the transaction network being ignored during training. Similarly, the presence of massive neighbor accounts further increases the difficulty of information extraction for target accounts.

Definition 12 (Mixing Data Graph): MDG is a directed graph $\mathcal{G}_D = (\mathcal{V}_C, \mathcal{V}_U, \mathcal{V}_N, \mathcal{E}_{CU}, \mathcal{E}_{UU}, \mathcal{E}_{UN}, \mathcal{E}_{DW})$, where $\mathcal{V}_C$, $\mathcal{V}_U$, $\mathcal{V}_N$ are the account sets of Tornado Cash mixing contracts, mixing users and their corresponding neighbors. $\mathcal{E}_{CU}$, $\mathcal{E}_{UU}$, and $\mathcal{E}_{UN}$ are three directed edge sets that reflect the transactions between mixing contracts and mixing users, the transactions between mixing users, as well as the transactions between mixing users and their relevant neighbors, where $\mathcal{E}_{CU} = \{(v_i, v_j, ts, tv, gl, gp, gu) | v_i \in \mathcal{V}_C, v_j \in \mathcal{V}_U\} \cup \{(v_i, v_j, ts, tv, gl, gp, gu) | v_i \in \mathcal{V}_U, v_j \in \mathcal{V}_C\}$, $\mathcal{E}_{UU} = \{(v_i, v_j, ts, tv, gl, gp, gu) | v_i, v_j \in \mathcal{V}_U\}$, $\mathcal{E}_{UN} = \{(v_i, v_j, ts, tv, gl, gp, gu) | v_i \in \mathcal{V}_N, v_j \in \mathcal{V}_U\} \cup \{(v_i, v_j, ts, tv, gl, gp, gu) | v_i \in \mathcal{V}_U, v_j \in \mathcal{V}_N\}$. $\mathcal{E}_{DW} = \{(v_i, v_j) | v_i, v_j \in \mathcal{V}_U\}$ is also a directed edge set, where each element refers to the association between deposit and withdrawal accounts of mixing users obtained in Section VI-B4.

To explore more potential features that help associate mixing user accounts, we transform MDG into uniformly sparse MTG through node merging and feature extraction. The key transaction information of mixing contracts and neighbor accounts is selectively derived and effectively merged into each mixing user account as part of the node features. Then, the Tornado Cash account nodes and their mixing transaction edges are removed from MDG to obtain an MTG containing only mixing accounts and their neighbor nodes. Table III shows the original node features with 100 dimensions, including four categories: pattern, quantity, time, and amount. After standardizing these features, we identify and delete redundant items based on the Pearson correlation coefficient and the variance inflation factor (VIF). Fig. 5 displays the heatmap of Pearson correlation coefficients between pattern features and other features. Principal component analysis (PCA) is applied to the remaining 75-dimensional features, ultimately outputting 32-dimensional features that represent the most informative aspects of the original features.

Definition 13 (Mixing Transfer Graph): MTG is a directed graph $\mathcal{G}_T = (\mathcal{V}_U, \mathcal{V}_N, \mathcal{E}_{UU}, \mathcal{E}_{UN})$, where $\mathcal{V}_U$ and $\mathcal{E}_{UU}$ represent the sets of mixing accounts and their associated edges. $\mathcal{V}_N$ and $\mathcal{E}_{UN}$ represent the sets of neighbor accounts and their transaction edges with mixing accounts, respectively. A and $\mathcal{X}_U$ denote the adjacency matrix and the feature matrix of $\mathcal{G}_T$, where $\mathcal{X}_U \in \mathbb{R}^{|\mathcal{V}_U| \times z}$ and z are denoted as the number of feature dimensions.

TABLE III
ORIGINAL ACCOUNT FEATURES AND THEIR INTERPRETATION

Categories	Descriptions	Features
Pattern	Number of total/0.1 ETH/1 ETH/10 ETH/100 ETH mixing transactions for six usage patterns	f_1 - f_{30}
Quantity	Number of total/deposit/withdrawal transactions for four mixing denominations (Ether)	f_{31} - f_{42}
	Number of deposit/withdrawal transactions	f_{43} - f_{44}
	Number of non-mixing transactions and neighbor accounts	f_{45} - f_{46}
	Out-degree and in-degree	f_{47} - f_{48}
Time	Timestamp of the first deposit/withdrawal/ non-mixing transaction	f_{49} - f_{51}
	Timestamp of the last deposit/withdrawal/ non-mixing transaction	f_{52} - f_{54}
	Time interval within deposit/withdrawal/non-mixing transactions (maximum, minimum, sum, mean)	f_{55} - f_{66}
Amount	Amount of deposit/withdrawal/non-mixing transactions (sum, mean)	f_{67} - f_{72}
	Amount of GasLimit/GasPrice/GasUsed for deposit/withdrawal/non-mixing transactions (maximum, minimum, mean)	f_{73} - f_{99}
	balance	f_{100}

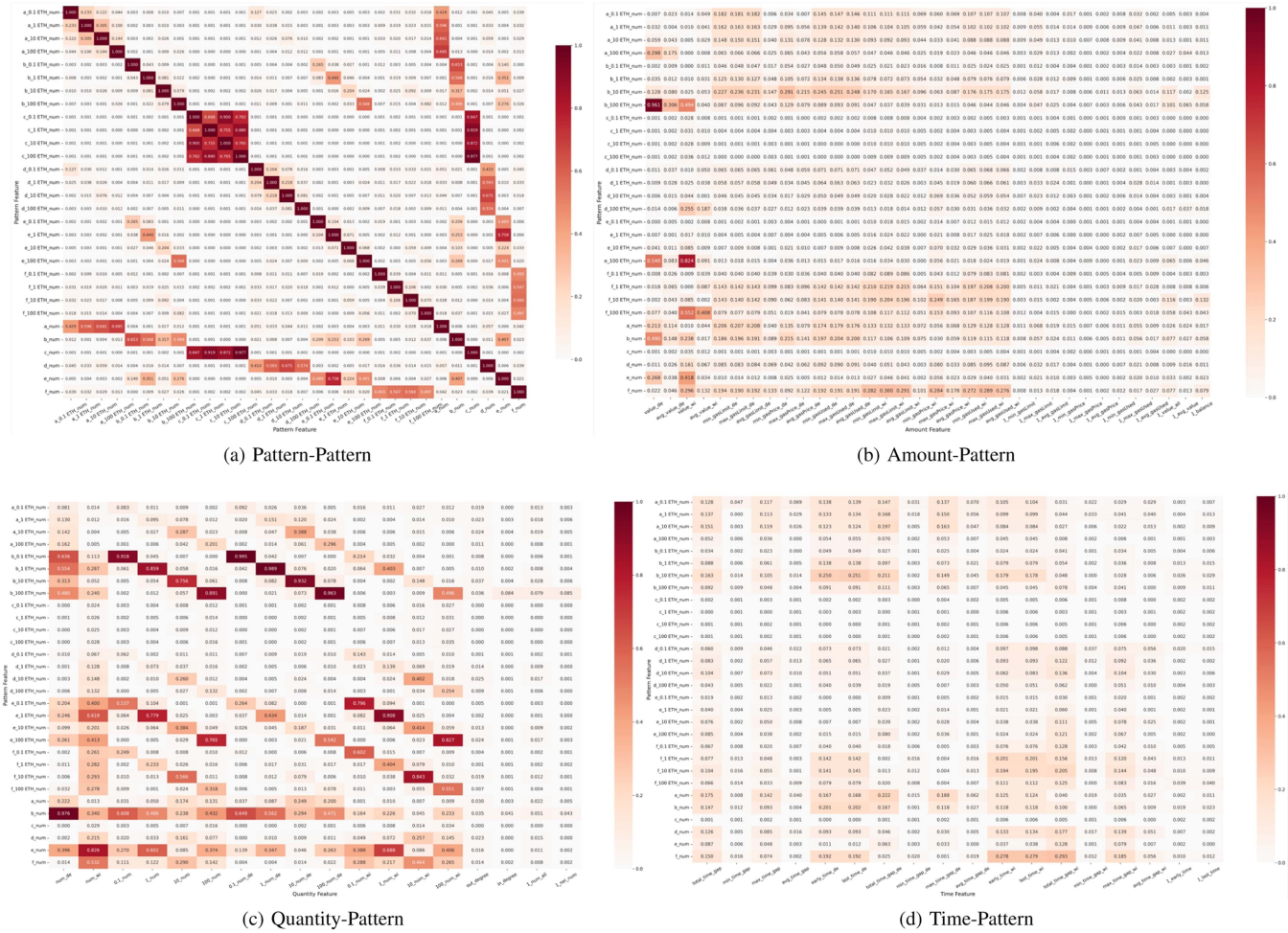
D. Mixing Accounts Correlation

After the previous data processing stages, the account correlation problem of Tornado Cash mixing users has been transformed into a link prediction task for graph node pairs in MTG. It attempts to speculate on the presence or absence of a connection edge between a pair of mixing user accounts at nodes, utilizing reliable information about the nodes and network structure in MTG. The mixing account correlation stage consists of subgraph sampling, structure encoding, and link decoding. Assume that a deposit account $addr_d$ and a withdrawal account $addr_w$ are randomly selected from the mixing user account set as the target node pair for analysis.

1) Subgraph Sampling: We sample a k-order enclosing subgraph for this node pair of deposit and withdrawal account nodes from the MTG, which covers the k-order subgraphs of each node and the edges between the two subgraphs. The enclosing subgraph offers several advantages over the regular subgraph. Firstly, the local structure around the node pair can be more completely revealed due to better connectivity insights, helping the model learn the relative positions and relationships between nodes. Secondly, it improves the robustness of the graph representation learning models by establishing a strong correlation between the target node and its regional subgraph, avoiding the impact of any potential disruptions within the subgraph itself. Finally, computational overhead can be reduced through sparsification techniques.

2) Information Encoding: This component encodes the pairwise and neighborhood structure of the target node pairs to learn the node representations and calculate their feature similarities.

Starting with the neighborhood structure, we intend to capture the structural information of the target node and learn its



(a) Pattern-Pattern (b) Amount-Pattern

(c) Quantity-Pattern (d) Time-Pattern

Fig. 5. Heatmap of Pearson correlation coefficients between pattern features and other features.

corresponding representations through the GNN model combined with the labeling trick. Considering that the GNN model is designed to learn a single-node representation, its use alone brings a fundamental restriction to our link prediction tasks. The common practice of directly aggregating the generated individual node representations into a joint representation for multiple nodes cannot capture the dependencies between the nodes in the target node set. The introduction of the labeling trick helps the GNN model learn the structural representation of the node set by aggregating the structural node representations obtained in the labeled graph [47]. Specifically, the DRNL technique [48] is selected to encode structural features. It assigns a unique integer label to each node based on the relative distance of the node from the target nodes in the enclosing subgraph, whose formula for the label $f_{nl}(addr_i)$ of an account node $addr_i$ is as follows:

$$f_{nl}(addr_i) = 1 + \min(d_{addr_d}, d_{addr_w}) + \left(\frac{d}{2}\right)\left[\left(\frac{d}{2}\right) + (d \bmod 2) - 1\right], \quad (7)$$

where d_{addr} denotes the shortest path distance between $addr_i$ and $addr$, $d = d_{addr_d} + d_{addr_w}$. Then, the GCN model [49] computes the node embedding vectors with the DRNL-encoded structural features as input, as shown below:

$$H^{l+1} = \sigma(\tilde{D}^{\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}), \quad (8)$$

where H^l and $W^{(l)}$ are the matrices of node embeddings and trainable parameters in the l-th layer, respectively. σ denotes the ReLU activation function. $\tilde{A}$ and $\tilde{D}$ represent the adjacency matrix and the degree matrix with self-loops added. $\tilde{A} = A + I$ and $\tilde{D}_{ii} = \sum_{j \in V} \tilde{A}_{ij}$, where I denotes the identity matrix. We concatenate the GCN-encoded structural representation with the additional feed-forward network (FFN)-encoded node feature representation, which is then passed through SortPooling to generate a similarity representation.

We adopt the binary structure transformer (BST) [50] to enhance the acquisition of the pairwise structure between the nodes. This model is more structure-aware than conventional transformer or GNN-based models, avoiding potential noise or information loss caused by nodes that are distant from the targets. The encoder of the pairwise structure consists of multiple independent BST layers, and a BST layer contains two sub-layers: binary structure attention (BSA) mechanism and a

simple positional fully connected feed-forward network. The two sub-layers are connected by a residual connection, followed by layer normalization. BSA is the core component of BST that takes the node features and pairwise structure features of the target deposit and withdrawal accounts as inputs. The node features have already been obtained during the MTG construction stage, while structural features are computed by various heuristics as follows:

– *Common Neighbor (CN):*

$$f_{CN} = |\Gamma(addr_d) \cap \Gamma(addr_w)|, \tag{9}$$

where $\Gamma(addr)$ is the set of neighbor accounts for *addr*.

– *Adamic-Adar index (AA):*

$$f_{AA} = \sum_{addr_i \in \Gamma(addr_d) \cap \Gamma(addr_w)} \frac{1}{\log(\deg(addr_i))}, \tag{10}$$

where $\Gamma(addr_d) \cap \Gamma(addr_w)$ is the set of common neighbor accounts for the target node pair. $\deg(addr)$ represents the degree of *addr*.

– *Jaccard index:*

$$f_{Jac} = \frac{|\Gamma(addr_d) \cap \Gamma(addr_w)|}{|\Gamma(addr_d) \cup \Gamma(addr_w)|}, \tag{11}$$

– Shortest path distance (SPD): length of the shortest path from $addr_d$ to $addr_w$, which is expressed as f_{SPD}.

– Shortest path number (SPN): number of shortest paths from $addr_d$ to $addr_w$, which is expressed as f_{SPN}.

We aggregate all heuristic structural vectors to form the query and key structural vectors, which are derived by performing the embedding lookup in the independent parameterized tables.

$$P_{Q(d,w)} = P^{CN}_{Q(d,w)} + P^{AA}_{Q(d,w)} + P^{Jac}_{Q(d,w)} + P^{SPD}_{Q(d,w)} + P^{SPN}_{Q(d,w)}, \tag{12}$$

$$P_{K(d,w)} = P^{CN}_{K(d,w)} + P^{AA}_{K(d,w)} + P^{Jac}_{K(d,w)} + P^{SPD}_{K(d,w)} + P^{SPN}_{K(d,w)}, \tag{13}$$

The structure encoding is the addition of these structure vectors, multiplied by two linear transformations of the target node features, respectively.

$$b_{dw} = q_d \cdot P_{Q(d,w)} + k_w \cdot P_{K(d,w)}, \tag{14}$$

where $q_d = h_d W^Q$ and $k_w = h_w W^K$ represent linear transformations of the target node features. W^Q and W^K denote the transformation matrices.

Finally, the single-head BSA can be calculated as:

$$\text{Attn}(H) = \text{softmax}\left(\frac{q_d \cdot k_w + b_{dw}}{\sqrt{d'}}\right) \cdot h_w W^V, \tag{15}$$

where d' is the dimension of q_d and $H = \{h_d, h_w\}$.

The multi-head BSA is implemented by performing the single-head BSA several times in parallel.

$$\text{MulAttn}(H) = \text{Concat}(\text{Attn}_1(H), \ldots, \text{Attn}_k(H)) W^O, \tag{16}$$

where W^O is a linear transformation matrix for multi-head information fusion, $H = \{h_d, h_w\}$ and Concat() represents the concatenate function.

3) Link Decoding: This component leverages the relational representation of the target node pair to calculate the probability of a link between them. All embeddings obtained in the previous component are concatenated to form a relational representation. The multi-layer perceptron (MLP) decodes the relational representation to generate a scalar, which is converted into a probability via a sigmoid function. This process is formulated as

$$h_{d,w} = \text{Concat}(h^{\text{GCN}}_{d,w} || h^{\text{BST}}_{d,w}), \tag{17}$$

$$h^{(1)} = \sigma(W^{(1)} h_{d,w} + b^{(1)}), \tag{18}$$

$$\tilde{y} = \text{sigmoid}(W^{(2)} h^{(1)} + b^{(2)}), \tag{19}$$

where $h^{\text{GCN}}_{d,w}$ is the similarity embedding of target nodes encoded by GNN and $h^{\text{BST}}_{d,w}$ is the relation embedding encoded by BST. Concat() represents the concatenate function.

VII. Experimental Evaluation

We perform a comprehensive evaluation of the proposed method, including account feature evaluation, advanced method comparison, and ablation analysis.

A. Experimental Settings

1) Datasets: We experimented with real transactions on the Ethereum mainnet from December 15, 2019, to August 8, 2022, during which Tornado Cash operated properly without sanctions. Approximately 817,500 raw Tornado Cash transactions were collected from the Ethereum client Geth, with internal and external transactions of 517,733 and 299,701, respectively, following the components of the proposed method in Section VI. Table IV shows the collection results of Tornado Cash core contracts. As seen in Table IV, 43,218 accounts initiated 153,073 deposit transactions, while only 4151 accounts created 138,278 withdrawal transactions. After the mixing data preparation stage, we derived a real mixing transaction dataset containing 272,236 records and 83,089 accounts, as well as a ground-truth dataset of 949 mixing account pairs.

2) Baselines: We select the following representative-related works to compare with the proposed method.

– *Beres et al. [25]:* three heuristic rules for linking deposit and withdrawal transactions, including mixing account reuse, unique gas price, and transaction-linked accounts.

– *Tang et al. [26]:* three heuristic rules based on the mixing transaction patterns of single deposit & withdrawal, multiple pairs of single deposit & withdrawal, and multiple deposits & withdrawals.

– *Hu et al. [36]:* a pre-trained transformer for various Ethereum fraud detection tasks. The method, equipped with strategies for repetitiveness reduction, skew alleviation, and modeling heterogeneity, can identify accounts with the same ownership in Tornado Cash.

– *Du et al. [29]:* a GNN-based link prediction model. It captures the interconnectivity of nodes within the mixing interaction graph and computes the association probability between account nodes via node embedding.

TABLE IV
RAW DATA COLLECTION RESULTS ON TORNADO CASH

Contracts	Tags	Active Period	Functions	Patterns	$\lvert\mathcal{T}\rvert^\star$	$\lvert\mathcal{T}'\rvert^\star$	$\lvert\mathcal{T}^d\rvert^\star$	$\lvert\mathcal{A}^d\rvert^\star$	$\lvert\mathcal{T}^w\rvert^\star$	$\lvert\mathcal{A}^w\rvert^\star$
0x12D66F...	Mixer 0.1 ETH	2019.12.16-2024.07.01	Deposit, Withdrawal	a, c, d	9,685	57,057	5,140	3,174	4,185	229
0x47CE0C...	Mixer 1 ETH	2019.12.16-2023.06.14	Deposit, Withdrawal	a, c, d	12,422	132,498	6,309	2,896	5,752	293
0x910CBD...	Mixer 10 ETH	2019.12.16-2024.07.01	Deposit, Withdrawal	a, c, d	13,860	119,709	7,006	2,494	6,612	392
0xA160CD...	Mixer 100 ETH	2019.12.25-2023.06.14	Deposit, Withdrawal	a, c, d	8,618	74,293	4,346	1,125	3,978	225
0x905B63...	Old Proxy	2020.12.18-2021.03.30	Withdrawal	e, f	37,217	22,304	21,933	5,392	14,884	703
0x722122...	Proxy	2021.03.31-2022.02.21	Withdrawal	e, f	143,903	67,412	71,390	17,000	67,875	1,557
0xD90E2f...	Router	2022.02.21-2024.07.01	Deposit, Withdrawal	b, e, f	73,996	44,460	36,949	9,751	3,4992	752
Totals					299,701	517,733	153,073	41,832	138,208	4,151

* $|\mathcal{T}|$, $|\mathcal{T}'|$ represent the number of Ethereum external and internal transactions on Tornado Cash. $|\mathcal{T}^d|$, $|\mathcal{T}^w|$ represent the number of Tornado Cash deposit and withdrawal transactions. $|\mathcal{A}^d|$, $|\mathcal{A}^w|$ represent the number of unique account addresses on Tornado Cash.

Note that the approaches [27], [28] mentioned in Section II are not considered for comparison. This is because the former aims to determine potential criminal accounts, while the latter is similar in principle to methods [25], [26] based on account reuse and transaction association.

3) Metrics: Three metrics are introduced to quantitatively evaluate the performance of different mixing account correlation methods.

- Precision (M1): $Precision = \frac{TP}{TP+FP}$
- Recall (M2): $Recall = \frac{TP}{TP+FN}$
- F1-Score (M3): $F1 = \frac{TP}{TP+0.5(FP+FN)}$

4) Implementation: We implement all the model's training and testing operations with Python 3.8 and Pytorch 1.7. The experiments are run on a machine with a 20-core Intel Core i7-14700KF CPU@3.4 GHz, Nvidia GeForce RTX4070 Ti GPU, 32 GB RAM, and 2 TB HDD.

Hyperparameters of the compared schemes based on heuristics or neural network models are mostly inherited from the original papers of each method. For Tang et al. 's method [26], the time interval of the single deposit & withdrawal pattern is 180 seconds. An eight-layer pre-trained transformer with two attention heads is extracted in [36], whose maximum sequence length is 100. The encoder in [29] selects a two-layer GraphSAGE model, which outperforms GCN and GAT in experiments under the 42-dimensional account node features. Our scheme deploys a three-layer GCN model with 128 hidden dimensions and a BST model with eight attention heads in the encoder to analyze a first-hop enclosing subgraph. Other parameters of our scheme follow the default parameters of the existing schemes. The batch size and training epoch are 32 and 50 for all compared methods, respectively. The K-fold cross validation is applied to train and test the model, where $K = 10$ and the ratio of training data to test data is 9:1.

B. Experimental Results

1) Account Features Evaluation: We explored the impact of account features with different dimensions on the mixing address correlation. Four types of account feature dimensions are set in the experiment: 32, 42, 51, and 100. The feature dimensions of our LASC are 32 after feature selection and optimization, and 100 without these processes, respectively. 51 dimensions are the result remaining after removing the pattern features and the non-mixing transaction features from the original features. MixBroker [29] uses the largest account feature dimension, 42, among all representative works. Table V gives the detailed performance of our model in different dimensions of account features. It can be seen that the proposed model achieves the best experimental results in the case of 32-dimensional account features. Compared with the features commonly used in existing work, introducing pattern and non-mixing transaction features significantly improves the effect of mixing account address correlation. Hence, we choose the proposed method equipped with 32-dimensional features for evaluation in subsequent experiments.

TABLE V
PERFORMANCE OF DIFFERENT FEATURE DIMENSIONS ON MIXING ACCOUNT CORRELATION

Metric	32-dim	42-dim	51-dim	100-dim
M1	**0.905±0.032**	0.772±0.121	0.791±0.089	0.886±0.065
M2	**0.841±0.063**	0.689±0.136	0.683±0.050	0.813±0.087
M3	**0.871±0.049**	0.728±0.045	0.733±0.067	0.842±0.025

TABLE VI
PERFORMANCE OF DIFFERENT METHODS ON MIXING ACCOUNT CORRELATION

Metric	Beres-Gas	Beres-Tx	Tang	Hu	Du	Ours
M1	0.894	**0.908**	0.699	0.695	0.832	0.905
M2	0.096	0.063	0.008	0.512	0.768	**0.841**
M3	0.173	0.118	0.016	0.590	0.798	**0.871**

2) Related Methods Comparison: We compare the proposed method with the state of the art to fully understand its performance. For the baseline methods described in Section VII-A2, we pick several heuristic rules from Beres et al. [25] and Tang et al. [26] for comparison. The mixing account reuse heuristic [25] is not discussed, since it only generates mixing transaction pairs with the same deposit and withdrawal accounts. The heuristic based on multiple pairs of single deposit & withdrawal pattern, or multiple deposits & withdrawals pattern [26] mainly analyzes the same accounts that initiate several deposit or withdrawal transactions, thus their deanonymization results are not considered in the comparison.

Table VI describes the performance of different schemes for mixing account correlation. Our method ranks third in precision (M1), slightly lower than two heuristic-based methods [25], [26],

TABLE VII
EFFICIENCY OF DIFFERENT METHODS ON MIXING ACCOUNT CORRELATION

Methods	Training time (s)	Testing time (s)
Beres-Gas	-	**32**
Beres-Tx	-	597642
Tang	-	97
Hu	35917	7518
Du	25608	6976
Ours	**21142**	5428

TABLE VIII
PERFORMANCE OF DIFFERENT MODULES ON MIXING ACCOUNT CORRELATION

Metric	non-feature	non-pairwise	non-neighbor	Ours
M1	0.791±0.089	0.827±0.057	0.867±0.024	**0.905±0.032**
M2	0.683±0.050	0.779±0.032	0.743±0.068	**0.841±0.063**
M3	0.733±0.067	0.802±0.044	0.799±0.049	**0.871±0.049**

but ahead of all the compared methods in recall (M2) and F1 score (M3). These precision-leading heuristics exploit the regular time intervals between deposit and withdrawal transactions to design a set of explicit rules that precisely define the mixing behavior in specific situations. This specialization enables them to achieve high correlation precision in mixing users with weak privacy awareness but cannot comprehensively cover all possible situations, especially those involving criminals who deliberately hide their behaviors.

In addition, we analyze the efficiency of different methods in terms of training time and testing time. Training time refers to the time required for the model to train, which includes the total time of all training epochs, while testing time refers to the time required for the model to infer all unseen data. Since heuristic-based methods do not need to be trained, the execution time of these algorithms is regarded as the testing time. From Table VII, it can be observed that our method outperforms all neural network-based methods in both training time and testing time. Although the heuristics based on gas price and single deposit & withdrawal pattern require little time for inference, their performance is significantly lower than our work.

3) Ablation Analysis: To examine the importance of each deanonymization strategy in LASC, we created three variants based on our complete linkability attack, which removes the introduction of pattern and non-mixing transaction features, neighborhood structure analysis, and pairwise structure analysis, respectively. The performance of these variants is shown in Table VIII. It can be seen that removing pattern and non-mixing transaction features leads to a drop in performance. This reaffirms the results discussed in Section VII-B1, that these features are effective in inferring mixing account correlations. The arbitrary absence of structural analysis components in the mixing account correlation module also degrades the performance of our model, indicating that they can capture the structural features missing from each other to improve the inference effect. Furthermore, the presented ENS heuristics increase the number of reliable account pairs from 103 to 949 compared with the latest dataset [29].

TABLE IX
TORNADO CASH TRANSACTION STATISTICS OF REAL MIXING USERS WITH FOUR DENOMINATIONS UNDER DIFFERENT USAGE PATTERNS

Patterns	0.1 ETH	1 ETH	10 ETH	100 ETH	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{A}\rvert$
$\mathcal{P}_a$	5140	6309	7006	4346	22801	7646
$\mathcal{P}_b$	18206	41803	34662	24838	119509	30741
$\mathcal{P}_c$	10	22	18	59	109	6
$\mathcal{P}_d$	4175	5730	6594	3919	20418	10176
$\mathcal{P}_e$	3165	7534	2675	5861	19235	2217
$\mathcal{P}_f$	11799	31148	30220	16997	90164	43226

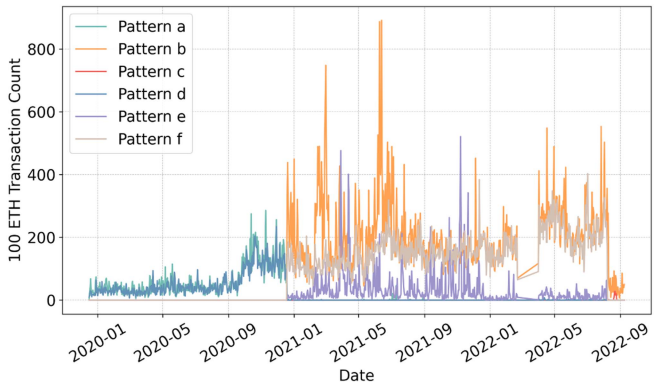


Fig. 6. Transaction Number of Different Patterns on Tornado Cash over Time (ETH).

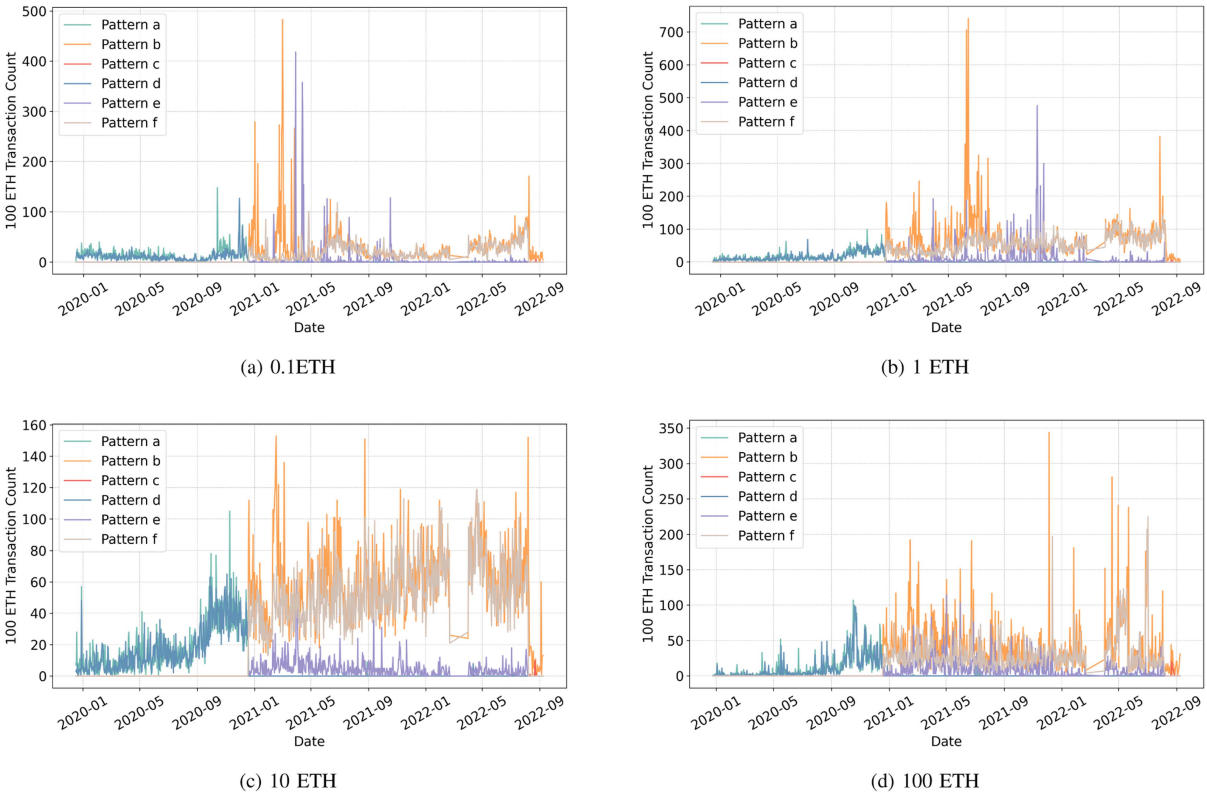
VIII. DISCUSSION

In this section, we discuss some deanonymization findings and performance improvements of LASC.

A. Findings

We conducted a statistical analysis of the number of transactions and addresses under four mixing denominations in different patterns (Table IX) and plotted the changes in the transactions of different patterns over time in Figs. 6 and 7. Combining the observed phenomena in the chart with the deanonymization results, we summarize the findings:

- During Tornado Cash's first year, users preferred deposits and withdrawals using patterns a and d. The mixing patterns and amounts between ordinary individual accounts and special accounts (not necessarily illicit) differed significantly, facilitating the association of the accounts for LASC.
- With the release of proxy and router contracts in early 2021, patterns b, e, and f became popular choices for mixing users. The correspondence between deposit and withdrawal patterns for different denominations has been complicated, thereby reducing the deanonymization effect. Especially with the mixing denomination of 10 ETH, there exists a huge transaction count gap between pattern e and pattern f.
- Tornado Cash's mixing transaction volume experienced significant dips in March, April, and August 2022, likely due to a decrease in public trust following regulatory penalties for mixing services.



(a) 0.1ETH (b) 1 ETH

(c) 10 ETH (d) 100 ETH

Fig. 7. Transaction Number of Different Patterns with Four Mixing Denominations on Tornado Cash over Time (ETH).

– Multiple transaction spikes with high denominations may be relevant to money laundering across various criminal activities.

B. Improvements

1) Timeliness: Despite the priority of accuracy over timeliness in anti-money laundering scenarios, enhancing the deanonymization method's performance has been a valuable improvement direction. In particular, the number of mixing accounts with their related transactions continues to increase over time, making the MTG scale larger and the degree distribution more heterogeneous. These factors will lead to a decrease in the accuracy and efficiency of the proposed method. Adjusting the model that encodes structural information and adopting appropriate graph optimization techniques are both worthwhile research directions to maintain high accuracy while achieving strong timeliness.

2) Generalization: While LASC demonstrates robust deanonymization capabilities for Tornado Cash users who mix ethers on Ethereum, its current scope presents two strategic expansion pathways. Tornado Cash operates on multiple ecosystems (e.g., Arbitrum, Binance Smart Chain, and Polygon) with cryptocurrencies like USDC and USDT. The LASC architecture can be adapted to these environments by developing chain-specific preprocessors for ledger formats or implementing denomination-aware feature extractors for the pools. Similar improvements are also applied to deanonymizing other SC-CMS schemes, such as Typhoon Cash [51] and Typhoon Network [52]. Moreover, our ground-truth dataset, while the largest academic collection in this field, exhibits inherent bias in ENS-based heuristics. Future work will expand the dataset by collaborating with regulatory entities or exploring potential on-chain trust services.

3) Interpretability: Although LASC performs well at associating mixing accounts, its GCN component reduces model interpretability in blockchain forensics, as false positives may carry significant legal implications. To address this gap, future work will integrate explainability techniques into LASC. While ensuring high efficiency and accuracy, the model will either replace the GCN component with an inherently interpretable architecture or leverage post-hoc interpretability methods such as GNNExplainer or attention visualization to transparently justify its outputs.

4) Ethical Considerations: LASC's design and implementation prioritize ethical integrity. All data derives from publicly accessible websites and blockchain, containing only anonymous transactions, without any real-world personal information. LASC is designed exclusively for forensic assistance in investigations of illegal activities. It outputs probabilistic linkages rather than deterministic accusations, requiring further verification by human analysts. Potential application risks, such as false positives and misuse, can be mitigated through confidence score reports, decision threshold adjustments, and access control for the model and the code.

IX. Conclusion

This paper focuses on designing a theoretically and practically feasible deanonymization technology for SC-CMS represented by Tornado Cash. We establish the first formal notion of SC-CMS through cryptographic syntax and security games, providing theoretical support for the systematic evaluation of mixing services and the effectiveness verification of deanonymization methods. Based on this notion, a linkability attack, LASC, is proposed to associate the mixing accounts of Tornado Cash, which has been proven to be feasible both in theory and practice. The attack derives six patterns of mixing implementation from Tornado Cash's mixing mechanism to restore the actual mixing accounts and infers the associations between mixing accounts in the constructed mixing transfer graph (MTG) through the enhanced graph structural learning model. A robust ground-truth dataset for Tornado Cash deanonymization is constructed, expanding the number of reliable account pairs by nearly an order of magnitude. Experiments demonstrate that LASC exhibits superior performance and efficiency compared with related work, paving the way for tracking money laundering activities through mixing services.

Acknowledgment

In addition, we express our sincere gratitude to anonymous reviewers for their valuable time and insightful comments, which greatly contributed to the quality and rigor of this paper. We also appreciate the editors for their professional handling of the manuscript and the constructive guidance they provided throughout the review process.

References

[1] Chainalysis Team, "2025 crypto crime report," Chainalysis, Tech. Rep., 2024. [Online]. Available: https://go.chainalysis.com/2025-Crypto-Crime-Report.html

[2] Chainalysis Team, "2024 crypto crime report," Chainalysis, Tech. Rep., 2023. [Online]. Available: https://go.chainalysis.com/crypto-crime-2024.html

[3] Chainalysis Team, "2023 crypto crime report," Chainalysis, Tech. Rep., 2022. [Online]. Available: https://go.chainalysis.com/2023-crypto-crime-report.html

[4] L. M. (FOX), "$625m ronin crypto heist: What to know," Mar. 2022. [Online]. Available: https://www.foxbusiness.com/markets/ronin-crypto-heist-what-to-know

[5] M. S. (CNBC), "More than $320 million stolen in latest apparent crypto hack," Feb. 2022. [Online]. Available: https://www.cnbc.com/2022/02/02/320-million-stolen-from-wormhole-bridge-linking-solana-and-ethereum.html

[6] K. N. (BBC), "Crypto theft: North korea-linked hackers stole $1.7b in 2022," Feb. 2023. [Online]. Available: https://www.bbc.com/news/world-asia-64494094

[7] K. J. B. (CBS), "Cryptocurrency heists are getting more ambitious — and costlier to investor," Apr. 2022. [Online]. Available: https://www.cbsnews.com/news/cryptocurrency-theft-hack-defi-beanstalk-blockchain/

[8] J. K. (CNN), "Another crypto bridge attack: Nomad loses $190 million in 'chaotic' hack," Aug. 2022. [Online]. Available: https://edition.cnn.com/2022/08/03/tech/crypto-bridge-hack-nomad/index.html

[9] T. TSIHITAS, "Worldwide cryptocurrency heists tracker (updated daily)," Apr. 2024. [Online]. Available: https://www.comparitech.com/crypto/biggest-cryptocurrency-heists/

[10] WIKIPEDIA, "Tornado cash," 2025. [Online]. Available: https://en.wikipedia.org/wiki/Tornado_Cash

[11] D. Lin et al., "Connector: Enhancing the traceability of decentralized bridge applications via automatic cross-chain transaction association," *IEEE Trans. Inf. Forensics Secur.*, vol. 20, pp. 7588–7601, 2025.

[12] F. (2012-2023), "International standards on combating money laundering and the financing of terrorism & proliferation," FATF, Paris, France, Tech. Rep., 2023. [Online]. Available: www.fatfgafi.org/en/publications/Fatfrecommendations/Fatf-recommendations.html

[13] Swift, "Know your customer (kyc)," Apr. 2024. [Online]. Available: https://www.swift.com/your-needs/financial-crime-cyber-security/know-your-customer-kyc

[14] W. E. Forum, "Pathways to crypto-asset regulation: A global approach," World Economic Forum, Geneva, Switzerland, Tech. Rep., May 2023. [Online]. Available: https://www.weforum.org/publications/pathways-to-crypto-asset-regulation-aglobal-approach/

[15] H. Mulhim, "Why crypto businesses need anti-money laundering regulations," Oct. 2022. [Online]. Available: https://www.weforum.org/agenda/2022/09/why-crypto-businesses-need-anti-money-laundering-regulations/

[16] D. Lin, J. Wu, Q. Fu, Z. Zheng, and T. Chen, "RiskProp: Account risk rating on ethereum via de-anonymous score and network propagation," *IEEE Trans. Dependable Secure Comput.*, vol. 22, no. 3, pp. 2132–2145, May/Jun. 2025.

[17] J. Liu, J. Chen, J. Wu, Z. Wu, J. Fang, and Z. Zheng, "Fishing for fraudsters: Uncovering ethereum phishing gangs with blockchain data," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 3038–3050, 2024.

[18] D. Williams, "U.S. v. storm and semenov indictment," Southern District of New York, United States District Court, New York, NY, USA, Tech. Rep., Sep. 2023. [Online]. Available: https://www.justice.gov/usao-sdny/file/1311391/dl?inline

[19] P.-P. A. E. Team, "Combatting illicit activity utilizing financial technologies and cryptocurrencies," Oct. 2022. [Online]. Available: https://www.dhs.gov/sites/default/files/2022-09/Combatting

[20] A. Gilbert, "Tornado cash founders agonised over kyc almost a year before sanctions, prosecutors say," Sep. 2023. [Online]. Available: https://www.dlnews.com/articles/regulation/tornado-cash-founders-agonised-over-kyc-prosecutors/

[21] T. Barbereau, E. Ermolaev, M. Brennecke, E. Hartwich, and J. Sedlmeir, "Beyond a fistful of tumblers: Toward a taxonomy of ethereum-based mixers," in *Proc. Int. Conf. Inf. Syst. Assoc. Inf. Syst.*, 2023, pp. 1780. [Online]. Available: https://aisel.aisnet.org/icis2023/cyber_security/cyber_security/13/

[22] A. Brownworth, J. Durfee, M. J. Lee, and A. Martin, "Regulating decentralized systems: Evidence from sanctions on tornado cash," *Federal Reserve Bank New York Staff Rep.*, vol. 8, pp. 1–38, 2024. [Online]. Available: https://www.newyorkfed.org/medialibrary/media/research/staff_reports/sr1112.pdf

[23] A. Pertsev, R. Semenov, and R. Storm, "Tornado cash privacy solution version 1.4," 2019. [Online]. Available: https://berkeley-defi.github.io/assets/material/Tornado%20Cash%20Whitepaper.pdf

[24] A. Wade, M. Lewellen, and P. V. Valkenburgh, "How does tornado cash work?," Sep. 2022. [Online]. Available: https://www.coincenter.org/education/advanced-topics/how-does-tornado-cash-work/

[25] F. Béres, I. A. Seres, A. A. Benczúr, and M. Quintyne-Collins, "Blockchain is watching you: Profiling and deanonymizing ethereum users," in *Proc. IEEE Int. Conf. Decentralized Appl. Infrastructures*, 2021, pp. 69–78.

[26] Y. Tang, C. Xu, C. Zhang, Y. Wu, and L. Zhu, "Analysis of address linkability in tornado cash on ethereum," in *Proc. 18th China Annu. Conf. Cyber Secur.*, 2021, vol. 1506, pp. 39–50.

[27] M. Youn, K. Chin, and K. Omote, "Empirical analysis of cryptocurrency mixer: Tornado cash," in *Proc. 2023 Congr. Comput. Sci., Comput. Eng., Appl. Comput.*, 2023, pp. 2324–2331.

[28] Z. Wang et al., "On how zero-knowledge proof blockchain mixers improve, and worsen user privacy," in *Proc. ACM Web Conf.*, 2023, pp. 2022–2032.

[29] H. Du, Z. Che, M. Shen, L. Zhu, and J. Hu, "Breaking the anonymity of ethereum mixing services using graph feature learning," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 616–631, 2024.

[30] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," in *Proc. IEEE Symp. Secur. Privacy*, 2015, pp. 104–121.

[31] Y. Hong, H. Kwon, J. Lee, and J. Hur, "A practical de-mixing algorithm for bitcoin mixing services," in *Proc. 2nd ACM Workshop Blockchains, Cryptocurrencies, Contracts*, 2018, pp. 15–20.

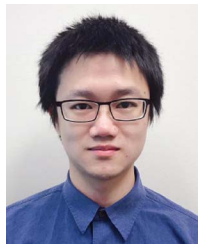
[32] J. Pakki, Y. Shoshitaishvili, R. Wang, T. Bao, and A. Doupé, "Everything you ever wanted to know about bitcoin mixers (but were afraid to ask)," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, 2021, vol. 12674, pp. 117–146.

[33] T. Tironsakkul, M. Maarek, A. Eross, and M. Just, "Tracking mixed bitcoins," in *Proc. Int. Workshops Data Privacy Manage., Cryptocurrencies Blockchain Technol.*, 2020, vol. 12484, pp. 447–457.

[34] L. Wu et al., "Towards understanding and demystifying bitcoin mixing services," in *Proc. Web Conf.*, 2021, pp. 33–44.
[35] C. Xu, R. Xiong, X. Shen, L. Zhu, and X. Zhang, "How to find a bitcoin mixer: A dual ensemble model for bitcoin mixing service detection," *IEEE Internet Things J.*, vol. 10, no. 19, pp. 17220–17230, Oct. 2023.
[36] S. Hu, Z. Zhang, B. Luo, S. Lu, B. He, and L. Liu, "BERT4ETH: A pre-trained transformer for ethereum fraud detection," in *Proc. ACM Web Conf.*, 2023, pp. 2189–2197.
[37] Groth, "On the size of pairing-based non-interactive arguments," in *Proc. 35th Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, M. Fischlin and J. Coron, Eds., 2016, vol. 9666, pp. 305–326.
[38] J. Baylina and M. Bellés, "4-bit window Pedersen hash on the baby jubjub elliptic curve," 2019. [Online]. Available: https://docs.iden3.io/publications/pdfs/Pedersen-Hash.pdf
[39] M. R. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen, "MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity," in *Proc. 22nd Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2016, vol. 10031, pp. 191–219.
[40] B. WhiteHat, J. Baylina, and M. Bellés, "Baby jubjub elliptic curve," 2019. [Online]. Available: https://docs.iden3.io/publications/pdfs/Baby-Jubjub.pdf
[41] P. S. L. M. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," in *Proc. 12th Int. Workshop Sel. Areas Cryptography*, 2005, vol. 3897, pp. 319–331.
[42] Y. Sakemi, T. Kobayashi, T. Saito, and R. S. Wahby, "Pairing-friendly curves," Crypto Forum Research Group, 2021. [Online]. Available: https://www.ietf.org/archive/id/draft-irtf-cfrg-pairing-friendly-curves-08.html
[43] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum & Parity Founder, 2024. [Online]. Available: [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf
[44] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Proc. Annu. Conf. Theory Appl. Cryptographic Techn.*, 1987, vol. 293, pp. 369–378.
[45] E. N. Service, "ENS documentation," 2024. [Online]. Available: https://docs.ens.domains/learn/protocol
[46] E. Medvedev and the D5 team, "Ethereum etl," 2018. [Online]. Available: https://github.com/blockchain-etl/ethereum-etl
[47] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2018, pp. 5171–5181.
[48] M. Zhang, P. Li, Y. Xia, K. Wang, and L. Jin, "Labeling trick: A theory of using graph neural networks for multi-node representation learning," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2021, pp. 9061–9073.
[49] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations (Poster)*, 2017, pp. 1–14. [Online]. Available: https://openreview.net/forum?id=SJU4ayYgl
[50] L. Shi, B. Hu, D. Zhao, J. He, Z. Zhang, and J. Zhou, "Structural information enhanced graph representation for link prediction," in *Proc. 38th AAAI Conf. Artif. Intell.*, 2024, pp. 14964–14972.
[51] Typhoon, "Typhoon cash," 2024. [Online]. Available: https://typhoon.cash/
[52] Typhoon, "Typhoon network," 2024. [Online]. Available: https://docs.typhoon.network/

Yan Wu (Member, IEEE) received the BE an ME degrees in engineering from the University of Electronic Science and Technology of China, in 2017 and 2020, respectively. She is currently working toward the PhD degree with the School of Cyberspace Science and Technology, Beijing Institute of Technology. Her research interests include blockchain, cryptographic protocols, and network security.

Cong Wu (Member, IEEE) received the BE degree from Xidian University and the PhD degree from Wuhan University. He was a research fellow with Nanyang Technological University, working with prof. Yang Liu. He is currently a postdoctoral researcher with The University of Hong Kong. His research focuses on the security and privacy of distributed intelligent systems. He is an associate editor for JISA, IJCS, and SPY.



Yebo Feng received the PhD degree in computer science from the University of Oregon, in 2023. He is currently a research fellow with the College of Computing and Data Science, Nanyang Technological University. His research interests include network security, blockchain security, and anomaly detection. He is the recipient of the Best Paper Award of 2019 IEEE CNS, Gurdeep Pall Graduate Student Fellowship of UO, and Ripple Research Fellowship.

Lin Li received the PhD degree in electronic science and technology from Beihang University, Beijing, China, in 2018. He is currently a senior engineer with CNCERT, Beijing. His current research interests include AI and network information security.

Xinyue Zhang is currently working toward the master's degree with the School of Cyberspace Science and Technology, Beijing Institute of Technology. Her research interests include blockchain and network security.

Zhen Li is currently working toward the PhD degree with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China. His research interests include privacy computing, AI security, and blockchain.

Jiahang Sun (Graduate Student Member, IEEE) is currently working toward the PhD degree with the School of Computer Science and Technology, Beijing Institute of Technology. His research interests include blockchain, anonymous communication, and network security.

Zijian Zhang (Senior Member, IEEE) is currently a professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology. His research interests include anonymous communication security, blockchain security, and AI security.

Jincheng An is currently an assistant research fellow with Qi An Xin Technology Group Inc. He is an expert in cybersecurity and technology standardizing document drafting. His research interests include cyber resilience theory and evaluation, and data security detection.

Zhitao Guan (Member, IEEE) is currently a professor with the School of Control and Computer Engineering, North China Electric Power University. His current research interests include blockchain, AI security, and privacy computing. He has authored more than 100 peer-reviewed journal and conference papers in these areas.

Yong Liu is currently a research fellow and a doctoral supervisor in cybersecurity technology with QiAnXin Technology Group Inc. His research interests include cloud computing security, vulnerability mining, and blockchain security evaluation.

Liehuang Zhu (Senior Member, IEEE) is currently a professor, secretary with the School of Cyberspace Science and Technology, Beijing Institute of Technology. He has authored or coauthored more than 100 peer-reviewed journal or conference papers, including 50+ IEEE/ACM Transactions papers. His research interests include security protocol analysis and design, wireless sensor networks, and cloud computing. He has been granted a number of IEEE Best Paper Awards, including IWQoS 17', TrustCom 18'.