Kosmosis: Crypto Rug Pull Detection and Prevention by Fusing On- and Off-Chain Data in a Knowledge Graph

Philipp Stangl* and Christoph P. Neumann†

*Department of Computer Science
Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany
e-mail: philipp.stangl@fau.de
†Department of Electrical Engineering, Media and Computer Science
Ostbayerische Technische Hochschule Amberg-Weiden, Amberg, Germany
e-mail: c.neumann@oth-aw.de

***Abstract*—Rug pulls have become a major threat to the integrity of blockchain ecosystems, with illicit activities surging and resulting in significant financial losses. Existing approaches to prevent rug pulls focus on transaction graph analysis within blockchain networks, but these methods are limited. We propose Kosmosis, an incremental knowledge graph construction approach that integrates semantically-enriched blockchain data with social media insights into a unified knowledge graph to identify and prevent rug pulls. We demonstrate how Kosmosis can extract semantic information from blockchain transactions using the application binary interface to decode smart contract interactions and tag addresses based on their extracted relationships. We provide a technical description of the knowledge graph construction process, highlighting key components, such as address relation extraction, tagging, and entity resolution. Our research aims to provide a more comprehensive understanding of blockchain ecosystems and contribute to the development of robust anti-fraud measures.**

Keywords-blockchain; knowledge graphs; cyber fraud; rug pull; security; smart contracts.

I. Introduction

Crypto assets use distributed ledger technology, like blockchain, as decentralized transaction ledger and for proof of ownership. Different types exist, each with unique roles: 1) Cryptocurrencies, like Bitcoin, function as digital currencies for storing or transferring value. 2) Fungible tokens are interchangeable tokens with various utilities in blockchain ecosystems, often crucial in Decentralized Finance (DeFi) protocols. 3) Non-Fungible Tokens (NFTs), in contrast, are unique digital assets proving ownership and authenticity, holding distinct values and cannot be exchanged on a one-to-one basis with other tokens.

In recent years, illicit activities in crypto have surged. Chainalysis reported a record $20.6 billion in illicit transactions in 2022 [1]. Since the rise of DeFi in 2020 and NFTs in 2021, rug pulls have become a major fraud scheme [2], threatening investors and integrity of the crypto asset sector.

The primary method for detecting fraudulent activity is transaction graph analysis within blockchain networks [34]. However, this approach has two key limitations. First, transacting parties are pseudonymous, with only their blockchain addresses publicly visible. Tracking an address is possible, but linking it to a real-world entity is challenging, as the analysis is restricted to observable blockchain data. Second, this method focuses only on asset type, quantity, and sender/receiver, ignoring transaction semantics, such as what happened in a transaction that caused the assets to get transferred, is not covered, thus, limiting the depth of analysis.

Knowledge Graphs (KGs) can integrate fragmented knowledge from diverse data sources, enabling semantic queryingand reasoning. They offer a holistic view for detecting fraud patterns in highly connected datasets [5]. A KG consists of uniquely identified entities and their semantical relations, structured ontologically. Their open-world assumption allows continuous data integration, enhancing crypto asset fraud analysis and fraud prediction.

The remainder of this paper provides a technical perspective on the Kosmosis approach to incremental KG construction, complementing its use case outlined in Section II in extension to [6]. The use case focuses on detecting and preventing rug pulls, a threat relevant across various blockchain platforms, with our prototype specifically targeting the Ethereum blockchain due to its widespread adoption. To support this objective, we first provide background information on the Ethereum blockchain and graph-based blockchain data mining methods in Section III. We then describe the Kosmosis approach to incremental KG construction in Section IV, emphasizing the pipeline that serves as the foundation for the detection phase of the use case. Finally, we outline future work in Section V and conclude the paper with a discussion of our findings.

II. Kosmosis Objectives & Use Case

To illustrate the vision of Kosmosis-enabled rug pull prevention methods, we described a hypothetical user story about homer_eth in [6]. The user story method of use case illustration was adopted from our previous work in [7]. Kosmosis aims to leverage a KG to enhance security in blockchain ecosystems by identifying and alerting users before they interact with fraudulent projects.

A. Over-Aching Objectives of Kosmosis

Objective 1: Identifying and Alerting Users of Rug Pulls. With rising crypto scams, Kosmosis seeks to integrate block-

 ISBN: 978-1-68558-258-6

chain data, social media, and other KGs into a unified KG. This facilitates semantic querying and reasoning, enabling the development of alerting methods based on cross-domain semantic analysis—where knowledge about on-chain behaviors and social media interactions can be correlated—to detect anomalous patterns and assign addresses with a risk score.

Objective 2: Incremental Construction of the KG. To maintain high data freshness, Kosmosis requires a pipeline for integrating updates without full reconstruction. This ensures the integration of the latest available information while preserving existing data.

Objective 3: Extracting Blockchain Transaction Semantics. Transaction graphs typically show asset transfers but lack semantic insights. Kosmosis extracts transaction semantics by decoding smart contract interactions using their Application Binary Interface (ABI), which aids in detecting sophisticated fraudulent behavior. At present, our prototype specifically targets the account-based transaction model, as implemented in Ethereum. Expanding the framework to other blockchains employing different accounting models, such as UTXO-based systems, is a future objective.

B. Summary of the Kosmosis Use Case of Rug Pull Prevention

In our position paper [6], a hypothetical user, Bob, who is relatively new to the NFT market, illustrates Kosmosis' potential for rug pull prevention. A rug pull is a scam where victims authorize fraudulent transactions. According to [2], rug pulls occur in five stages: 1) project creation, 2) pre-mint hype, 3) token price setting, 4) accumulation of capital, and 5) exit scam. The use case is based on the true story of the threat actor Homer_eth, an NFT creator and 𝕏 user, who launched his first NFT collection, *Ether Bananas*, followed by the release of *Ether Monkeys* and *Zombie Monkeys*.

Of these three NFT collections, *Ether Monkeys* created the biggest medial buzz, because it promised additional utility through a casino to gamble and a decentralized autonomous organization to govern the NFTs, according to [8]. This buzz draws Bob into the fray. Bob bought his first NFT from Homer_eth and became an active participant in Homer_eth's growing community. Bob's involvement in the community deepened over time. He engaged in discussions, shared his excitement with fellow members, and earned himself a whitelist spot that allows Bob to mint the upcoming NFT project *Ether (ETH) Banana Chips* by Homer_eth. Convinced of its potential, Bob minted the NFT when the opportunity arose, unaware of the underlying risks associated with his investment.

Unbeknownst to Bob, the proceeds from the mint were not locked within the smart contract for future development, as initially promised. Instead, these funds were directly transferred to the deployer address associated with Homer_eth. Subsequently, Homer_eth either redirected these proceeds to a new deployer address—potentially to facilitate a future fraudulent scheme, or transferred them to an exchange to realize profits from previous deceptive activities. Following the launch of *ETH Banana Chips*, the community experienced a prolonged period of uncertainty, marked by an absence of updates or communication from Homer_eth. For several months, no new developments were reported, leaving stakeholders uncertain about the project's trajectory. It was not until March 2022 that Homer_eth resurfaced, announcing a final NFT project, titled *Froggy Frens*. However, due to backlash from the community, Homer_eth deleted his 𝕏 account and vanished [8].

C. Kosmosis Extension

Kosmosis identifies potential rug pulls by semantically analyzing transaction patterns encoded within smart contract interactions and cross-referencing blockchain addresses with real-world entity data from social media and other external sources. Our approach is grounded in the assumption that scammers publicly disclose or explicitly link blockchain addresses in their social media posts to promote their scams. This linkage is crucial for Kosmosis, as it provides the primary method of associating blockchain transactions with social identities, which enhances the semantic richness of the constructed KG.

The detection logic within the KG evaluates transactions that involve high-risk state changes, such as bulk asset transfers shortly after a token mint event, and assigns risk scores based on the presence of correlated indicators (e.g., rapid withdrawal to external accounts controlled by the deployer). These risk scores can trigger automated alerts before submitting a new transaction, providing timely warnings to users. Had Bob used Kosmosis, it would have analyzed the transaction history prior to submitting his mint transaction to Ethereum. The system would have issued a rug pull warning based on patterns of fund diversion to deployer addresses.

III. Background

This section covers background on rug pulls and blockchain technology, with a particular focus on the Ethereum blockchain, as detailed in Section III-A. Following the blockchain aspects, we discuss related graph-based approaches for blockchain data mining in Section III-B. On the social media aspects of Kosmosis, our prior work includes correlating Reddit data with traditional stock market trends [9] and analyzing Twitter/𝕏 data using SPARQL [10].

A. The Ethereum Blockchain

Blockchain technology is founded on the principles of immutability, decentralization, transparency, and cryptographic security, and it has been applied across various domains in recent years. For example, it has been utilized in the financial sector (e.g., [11]), as well as in supply chain management, either through a single blockchain [12] or by leveraging multiple interoperable blockchains [13]. A significant subset of blockchain technology is smart contract platforms, which facilitate the development of decentralized applications through self-executing smart contracts. This section provides an overview of the key concepts of Ethereum as a representative smart contract platform. It covers fundamental aspects such as smart contracts, their execution environment, and the account-based transaction model, which are essential for the subsequent sections.

1) Blockchain Data Structure: A blockchain is a data structure whose elements called blocks are linked together to form a chain of blocks [14]. Each block comprises two parts: a body and a header. The body of the block contains a set of transactions. A transaction typically involves the transfer of assets between a sender and a receiver. These participants are represented by addresses, which are unique alphanumeric strings that clearly specify the origin and destination of each transaction. Further, the block body is used to generate a unique identifier called the block hash. The block header contains a reference to the unique identifier of its immediate predecessor, known as the parent block.

2) Smart Contracts: Through smart contacts, which are executable source codes that enforce the terms and conditions of particular agreements, a smart contract platform like Ethereum facilitates the development of decentralized applications [15]. Once deployed on the blockchain, the smart contract is assigned an address where the code resides and cannot be altered or tampered with. By writing custom smart contracts, developers can create and manage tokens that adhere to the standards ERC-20 for Fungible Token (FT) or ERC-721 for NFT. An ABI specifies the functions and data structures exposed by a smart contract, allowing external applications to understand the capabilities of the contract. Further, an ABI defines a format for encoding and decoding data that is passed between smart contracts and external applications. This ensures a consistent and standardized way to exchange information.

The Ethereum blockchain manages ETH as the native cryptocurrency of the platform. It operates with the Ethereum Virtual Machine (EVM) as a fundamental building block, serving as the execution environment for smart contract code. Smart contracts, primarily written in a high-level language such as Solidity, undergo compilation into EVM bytecode. This bytecode is the executable format used by the EVM to enact smart contract functions. To interact with this bytecode, a contract ABI is utilized, which acts as a bridge between the high-level language and the low-level bytecode. In this context, an EVM disassembler plays a crucial role; it reverses the bytecode back into a more readable format, aiding developers in understanding and analyzing the code deployed on the Ethereum blockchain. Figure 1 shows the processes involved in deploying smart contracts to the Ethereum blockchain and reading contract data from it, including compilation and deployment steps, and the interaction between a web application and the Ethereum blockchain. The left side shows the compilation and deployment of a smart contract, and the right side depicts an interaction with the contract (e.g., from a web application).

3) Externally Owned Account: Unlike smart contracts, Externally Owned Accounts (EOAs) are controlled by real-world entitys through private keys, enabling them to initiate transactions, such as transferring crypto assets or executing functions of a smart contract. When an EOA sends a transaction to a smart contract, it triggers the code of the contract to execute according to its predefined rules.

4) Account-based Accounting: For the record-keeping of transactions, blockchains utilize an accounting model. Compared to other blockchains, such as the equally well-known Bitcoin blockchain that uses the Unspent Transaction Output (UTXO) model, or its successor the extended UTXO [17] utilized by the Cardano blockchain, whereas Ethereum employs the account-based accounting model.

The account-based model can be best understood through the analogy of a bank account. This approach mirrors how a banking account operates. Like a bank account that tracks the inflow and outflow of funds, thereby reflecting the current balance, the account-based model in Ethereum maintains a state that records the balance of Ether. Thus, it is inherently stateful. Each transaction results in a direct adjustment to this balance, akin to a deposit or withdrawal in a bank account. This model's stateful nature ensures that at any given moment, the system can accurately reflect the total amount of Ether held in each account, offering an up-to-date view of account balances within Ethereum.

5) Token Minting: Token minting refers to the process of generating new tokens. Fungible tokens are typically minted by their creator either at the project's launch or gradually over time. This issuance is governed by predefined rules or algorithms embedded within the project's smart contracts.

The minting of NFTs involves participants other than the original token creator, commonly known as token minters. These individuals engage in the process by invoking a specific function within a smart contract, designated as `mint` in the ERC-721 token standard. Executing this function results in an increase in the total supply of NFTs while simultaneously assigning the newly minted tokens to the blockchain address of the minter.

The minting process for NFTs is frequently facilitated through a dedicated minting platform. Prospective minters or investors must contribute a predefined amount, as determined by the creator, to initiate the minting process. This contribution enables them to mint one or multiple NFTs, depending on the stipulations outlined in the smart contract. Beyond enabling the creation of new NFTs, this process also serves as a mechanism for directly transferring ownership from the NFT creator to the NFT minter.

B. Rug Pull Detection Methods

This section examines two main approaches used for rug pull detection: smart contract code analysis and graph-based methods. Smart contract code analysis entails a comprehensive examination of a contract's source code to extract and interpret the semantic behavior of transactions. For instance, [18] leverages this approach to uncover potential vulnerabilities and fraudulent patterns within smart contracts. Their proposed method, "Tokeer," systematically dissects the code to identify suspicious patterns and functions that may indicate a predisposition to rug pull schemes.

Graph-based techniques, on the other hand, leverage graph theory and data mining to analyze blockchain network graphs, as blockchain transactions naturally form graphs [19]. Elmougy

and Liu [20] describe three graph types for blockchain networks: *money flow transaction graphs* (representing how asset flow over time), *address-transaction graphs* (showing asset flow across transactions and addresses), and *user entity graphs* (clustering addresses potentially controlled by the same user to deanonymize them). Graph-based rug pull detection often uses network embedding techniques, such as graph convolutional networks (e.g., [21]), to automatically extract features from the blockchain network.

IV. The Kosmosis Approach to Incremental Knowledge Graph Construction

To incrementally construct a KG that integrates data in a continuous and periodic way, we propose a multi-stage pipeline, as illustrated in Figure 2. It originated from a master's thesis [22] and consists of three stages: Data ingestion, data processing, and knowledge storage. We use italics to emphasize on conceptual aspects and typewriter text for technical operations.

The initial stage, data ingestion, captures the raw data from the primary data sources (blockchain and social media) as well as enrichment data sources (e.g., another knowledge base). This phase is characterized by its versatility in the frequency of data acquisition: it can be 1) *continuous*, to capture real-time updates from sources such as blockchain nodes, 2) *incremental* for new posts via the 𝕏 Streaming Application Programming Interface (API), 3) *periodic*, to capture new entries in structured data sources like relational databases at regular intervals, or 4) *event-based*, responding to events that are emitted upon new entity additions to the KG.

Following the ingestion stage, the data processing stage is initiated, which is partitioned into distinct workflows tailored to handle each type of ingested data. This segmentation allows for specialized processing depending on the structure of the raw data. For instance, for text data sources, natural language processing techniques, such as named entity recognition [23], can be used to ensure that the data is accurately interpreted, and contextual relationships are discerned.

In the third and final stage, the refined data is loaded into the knowledge storage, where it is systematically organized within a triplestore, a type of database optimized for storing and retrieving data in Resource Description Framework (RDF) format. The triplestore can then be used for semantic querying capabilities to extract actionable insights from the KG for downstream processes. For the KG, we use the EthOn [24] ontology that formalizes the concepts and relations within the domain of the Ethereum network and blockchain. EthOn is written in RDF and Web Ontology Language (OWL).

A. Blockchain Data Processing

The blockchain data processing workflow continuously ingests new transactions from the blockchain via websocket connections. Websockets enable open, interactive communication sessions between a client and a server, facilitating real-time data transfer without the need for repeated polling. Upon receiving these transactions, the workflow processes and integrates them into the KG by first extracting the address relationship, followed by tagging the addresses, and finally fusing the addresses with the entities of the KG.

1) Address Relation Extraction: In order to provide answers to "why" and "how" assets were transferred in a transaction, Kosmosis implements a pipeline module titled *Address Relation Extraction*. The responsibility of this module is to extract the semantic information in a blockchain transaction through decoding the input data of a transaction using the ABI of the smart contract a blockchain address is interacting with.

First, the ABI is requested from Etherscan [25] and Sourcify [26] via their respective REST APIs. If the ABI cannot be successfully fetched from one of the aforementioned sources, the module resorts to reconstructing the ABI from the smart contract byte code, which is available at any time since the bytecode is deployed on the blockchain. This operation enables

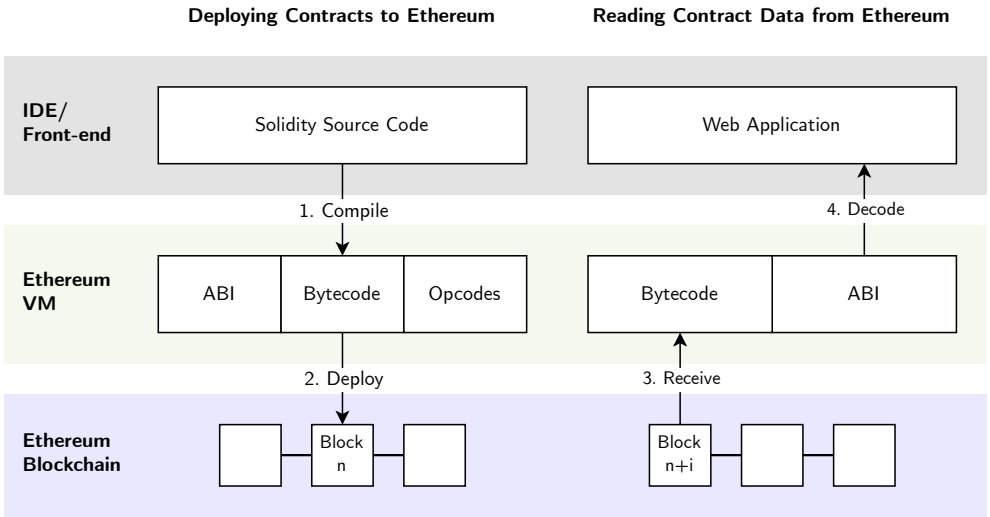

Figure 1. Schematic representation of deploying and reading from smart contracts. Adapted from Takeuchi [16].

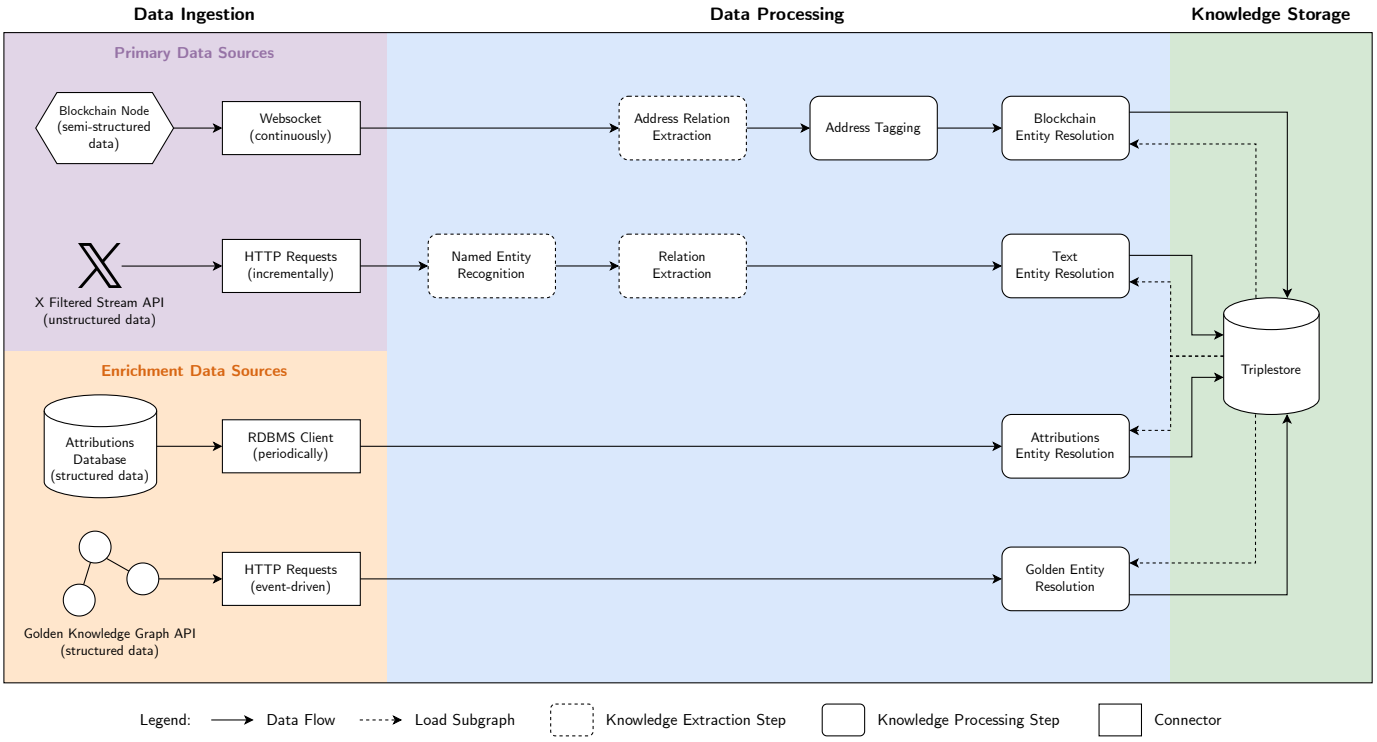

Figure 2. A high level overview of the Kosmosis pipeline.

the decoding of transactions and the interaction with smart contracts beyond their compiled state.

The initial step involves the disassembly of the bytecode of the smart contract. This operation, referred to as `DISASM`, decomposes the bytecode into a series of readable opcodes and associated data. Disassemblers (e.g., *pyevmasm* [27]) facilitate this step by translating the bytecode back into a form that represents the original instructions and operations defined within the smart contract.

Following disassembly, the algorithm initializes by creating an empty array intended to store the ABI and defining lists of opcodes that either change the state or read from the state of the blockchain. These opcodes include `SSTORE`, `CREATE`, `CREATE2` for state-changing operations, and `SLOAD` for state-reading operations, reflecting the fundamental actions a smart contract on the EVM can perform [11].

The core of the algorithm iterates over selector/offset pairs within the disassembled bytecode. Selectors serve as identifiers for functions in the EVM, facilitating the mapping to the corresponding functionality. If a given offset does not match any destination within the program's destinations, the iteration skips to the next pair, ensuring only valid functions are considered.

Upon finding a valid function destination, the algorithm retrieves the function definition and assigns tags based on its behavior. This tagging process involves analyzing the opcodes contained within the function and any related jump destinations. The purpose is to categorize functions according to how they alter the blockchain state, using a depth-first search algorithm to navigate through the function call graph.

An `AbiFunction` object is then created for each valid function, with its payable status determined inversely by the presence of a notPayable marker at the corresponding offset. The algorithm next assigns mutability attributes (nonpayable, payable, view, or pure) based on whether the function alters state, reads state, or neither. This classification is crucial for understanding how functions interact with the blockchain and their implications on transaction costs and permissions.

Finally, the algorithm decides on the inclusion of inputs and outputs in the function signature, informed by the presence of specific tags. For instance, tags indicating data retrieval or state mutation influence whether parameters are classified as inputs or outputs. This granular control ensures that the ABI accurately reflects the interface of the smart contract, allowing for effective transaction decoding.

Currently, the method for extracting semantic information from smart contract transactions relies partly on predefined heuristics, such as recognizing specific function names like "mint." However, we acknowledge that scammers could circumvent these simplistic heuristics by obfuscating or renaming functions. Future improvements will incorporate advanced transaction pattern analysis rather than function naming alone, enhancing resilience against simple obfuscation techniques.

2) Address Tagging: Since the exact identity of a real-world entity controlling a blockchain address is often unknown, it can still be categorized and tagged accordingly. The *address tagging module* tags the sender and receiver address based on their extracted relationship from the preceding address relation extraction module. For instance, an EOA deploying a smart

contract is tagged as deployer in case of a contract creation transaction. Likewise, if an EOA is sending Ether to an NFT contract T via a contract function containing the word "mint," the EOA is tagged as is tagged as NFT minter of T. Tags are subclasses of EOAs and contract accounts, extending the address concept of the EthOn ontology.

3) Blockchain Entity Resolution: The blockchain entity resolution module is responsible for resolving blockchain addresses to either new entities or existing ones in the KG, by using the extracted information from preceding steps. It begins with mapping the result data from the preceding steps into the RDF format, adhering to the ontology defined by the KG. This ensures that the data is structured in a way that is compatible with the KG's existing schema.

Following the mapping to RDF, the next phase involves fusing this RDF data with the KG. This is accomplished through a two-step process. Initially, a subgraph that is relevant to the processed data is loaded into the system. This step, commonly referred to as "blocking," narrows down the scope of the resolution process to the most relevant segments of the KG, thereby enhancing the entity resolution process.

Subsequently, the system proceeds to match the newly processed data with the corresponding entities within the KG. This matching process is crucial for identifying where the new data fits within the existing structure and for ensuring that it is integrated in a meaningful way. In certain cases, the fusion process may also involve the clustering of entities. This is particularly relevant for blockchain data, where unique characteristics of the data can be leveraged to enhance the integration process.

For instance, when dealing with blockchains that utilize an account-based accounting model, address clustering heuristics can be employed to further refine the fusion process. One such heuristic is the deposit address reuse, as proposed by Victor [28]. Kosmosis uses deposit address reuse for blockchain data from Ethereum to resolve entities more effectively.

B. Text Processing

The workflow starts with the input of unstructured data from the 𝕏 Filtered Stream API [29], which is incrementally streamed and parsed via a long-lived HTTP request into the pipeline. The first step in processing this data is named entity recognition, where the system identifies and classifies named entities present in the text into predefined categories, such as the names of persons, organizations, and locations.

The next step is relation extraction. This process involves identifying and extracting relationships between the named entities that were previously recognized. For instance, it could determine that a person named "Alice" works for a company named "Acme."

The final step in the text processing workflow is the entity resolution, achieved through blocking and matching. For each new entity, the system identifies all other entities within the KG that need to be considered for matching. Considering the growing size of the KG, through the incremental updates, it is important to limit the matching process to as few candidates as possible [30]. The method of limiting candidates is known as blocking, which confines the matching process to entities of the same or most similar entity type.

Following the blocking that serves as a preliminary filtering step, the matching is performed. This involves a pairwise comparison of the new entities with those existing entities in the KG identified during the blocking phase. Its objective is to identify all entities that are sufficiently similar and, therefore, potential candidates for matching. This pairwise comparison relies on a nuanced assessment of similarity that encompasses both the properties of the entities and their relational connections within the KG. By evaluating both property values and the nature of relationships to other entities, the system determines the degree of similarity between entities.

C. Enrichment Data Processing

Enrichment data enhances the data obtained from primary data sources with supplementary context regarding real-world entitys. Attributions involve the mapping of blockchain addresses to their corresponding real-world entities. This task is largely dependent on data sourced from a network of experts, such as team members from blockchain projects. The input data for the attribution process is typically not consistent in its timing, as it depends on when the experts provide updates or when new information becomes available. As a result, the enrichment data processing workflow is designed to operate at regular intervals, ensuring that the KG is updated systematically and remains as up-to-date as possible.

To further enrich the KG, data from external knowledge bases is integrated. In our case, we use the *Golden Knowledge Graph* due to its concentrated information on tech startups and cryptocurrencies. This external graph offers a wealth of information about crypto projects, including details about their founders, team members, and project descriptions. Such depth of data provides a valuable context that can significantly improve the understanding of entities in the constructed KG.

The workflow for integrating knowledge from an external KG is event-driven, activated once the knowledge storage indicates the addition of new entities from the social media platform 𝕏. Then, the workflow triggers a process to pull in additional background information from the Golden Enrichment API [31]. It uses the 𝕏 username that has been newly included in the KG as unique identifier to fetch relevant data.

D. Quantity Structure of the Knowledge Graph Data

In our prototype implementation, data was ingested at rates averaging 10-15 transactions per second (each averaging 5KB) from Ethereum blockchain nodes and roughly 200 tweets per minute (each averaging 2KB) from the X filtered stream API. This combined ingestion rate corresponds approximately between 3.4 to 4.9 MB per minute of raw data. Our prototype runs on a standalone cloud server instance with 32 GiB RAM and 8 vCPUs (AWS EC2 m5.2xlarge) with a 512GB SSD, managing real-time data ingestion and processing workloads. The semantic enrichment introduces minimal latency (less

than 5 seconds per transaction batch), thus allowing for near-real-time KG updates. The KG constructed by Kosmosis accumulates triples at an approximate rate of 2.5 to 6 million triples per day, depending on transaction activity and the level of detail extracted from social media.

While the described hardware configuration proved adequate for prototype-level or small- to medium-scale deployments, a production implementation aimed at analyzing multiple blockchain networks or higher data volumes would necessitate scaling to multiple compute nodes, each handling dedicated tasks such as blockchain data ingestion.

V. Conclusion and Future Work

In this paper, we demonstrated how to build a knowledge graph from blockchain and social media data using the Kosmosis approach to incremental knowledge graph construction. It complements our previous use case paper [6] that provided a real-world example of how Kosmosis can detect fraudulent activity, with a high-level technical discussion about the Kosmosis pipeline.

In the exemplary scenario, a threat actor known as Homer_eth executed five NFT project heists within two months, accumulating over $2.8 million in profits. We summarized our user story, in which Kosmosis provides a knowledge graph that improves the detection of such fraudulent schemes. Kosmosis fuses on- and off-chain data, thus, it becomes the basis for semantic querying and reasoning over a graph of entities and the relationships among them, facilitating analyses for cybercrime and fraud prevention, with the current focus on rug pulls as a major fraud scheme.

The initial findings of our research on Kosmosis have shown promising results, indicating the potential of our approach in identifying and preventing rug pulls. However, there are ample improvement opportunities for Kosmosis in future work.

It will be necessary to refine the filters used in the ingestion of data from the 𝕏 Filtered Stream API. The current process of data ingestion depends on the presence of direct links to blockchain addresses in social media posts. For instance, the ability to link the user Homer_eth with the *EtherReapers* smart contract was solely facilitated by the explicit mention of the smart contract address in Homer_eth's announcement post on 𝕏. This example underscores the limitations of the current approach, which may overlook relevant connections in the absence of direct references. Consequently, a more sophisticated approach is required to ensure a broader and still relevant dataset is captured to associate 𝕏 users with their respective blockchain addresses.

Additionally, the implementation of knowledge fusion, the process of identifying true subject-predicate-object triples [32], sourced from the blockchain and social media stands out as a critical next step. By fusing multiple records representing the same real-world entity into a single and consistent representation [33], knowledge fusion would allow for a more accurate representation of real-world entitys in the knowledge graph.

Currently, our prototype is limited to blockchains utilizing the account-based accounting model, like Ethereum. Recognizing the diversity in blockchain architectures and their unique features, we aim to allow for the integration of blockchains using a different accounting system, like Bitcoin. This expansion is essential for broadening the applicability and utility of Kosmosis across different blockchain platforms.

In conclusion, the Kosmosis pipeline supports the ingestion of unstructured, semi-structured, and structured data, as well as the ingestion of new data at different time intervals. It supports continuous ingestion in a stream-like fashion, incrementally, periodically, or event-based ingestion. During construction, the semantics of blockchain transactions are extracted to address "why" and "how" crypto assets were transferred.

References

[1] Chainalysis, "The 2023 crypto crime report," Chainalysis, Feb. 2023, [Online]. Available: https://go.chainalysis.com/2023-crypto-crime-report.html (visited on 01/31/2025).

[2] T. Sharma, R. Agarwal, and S. K. Shukla, "Understanding rug pulls: An in-depth behavioral analysis of fraudulent nft creators," *ACM Trans. Web*, vol. 18, no. 1, Oct. 2023, ISSN: 1559-1131. DOI: 10.1145/3623376.

[3] A. Khan, "Graph analysis of the ethereum blockchain data: A survey of datasets, methods, and future work," in *2022 IEEE International Conference on Blockchain (Blockchain)*, IEEE, Espoo, Finland: IEEE, 2022, pp. 250–257.

[4] F. Béres, I. A. Seres, A. A. Benczúr, and M. Quintyne-Collins, "Blockchain is watching you: Profiling and deanonymizing ethereum users," in *2021 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, Online: IEEE, 2021, pp. 69–78. DOI: 10.1109/DAPPS52256.2021.00013.

[5] X. Zhu *et al.*, "Intelligent financial fraud detection practices in post-pandemic era," *The Innovation*, vol. 2, no. 4, 2021.

[6] P. Stangl and C. P. Neumann, "The Kosmosis Use Case of Crypto Rug Pull Prevention by an Incrementally Constructed Knowledge Graph," in *Proc of the 2nd Workshop on Data Engineering for Data Science (DE4DS) in conjunction with the 21st Conference on Database Systems for Business, Technology and Web (BTW'25)*, Bamberg, DE, Mar. 2025, forthcoming.

[7] C. P. Neumann and R. Lenz, "The alpha-Flow Use-Case of Breast Cancer Treatment – Modeling Inter-Institutional Healthcare Workflows by Active Documents," in *Proc of the 19th Int'l Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2010)*, Larissa, GR, Jun. 2010, pp. 12–22. DOI: 10.1109/WETICE.2010.8.

[8] ZachXBT [@zachxbt], "Homer.eth (formerly @homer_eth) rug pull analysis," X, X Corp., May 26, 2022, [Online]. Available: https://x.com/zachxbt/status/1529973318563946496 (visited on 12/05/2023).

[9] T. Bauer *et al.*, "Reddiment: Eine SvelteKit- und ElasticSearch-basierte Reddit Sentiment-Analyse," German, Ostbayerische Technische Hochschule Amberg-Weiden, Technische Berichte CL-2022-06, Jul. 2022. DOI: 10.13140/RG.2.2.32244.12161.

[10] B. Hahn *et al.*, "Twitter-Dash: React- und .NET-basierte Trend- und Sentiment-Analysen," German, Ostbayerische Technische Hochschule Amberg-Weiden, Technische Berichte CL-2022-07, Jul. 2022. DOI: 10.13140/RG.2.2.15466.90564.

[11] G. Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger," (Ethereum project yellow paper), Parity Technologies, 2024, [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf (visited on 01/29/2024).

[12] S. Wang, D. Li, Y. Zhang, and J. Chen, "Smart contract-based product traceability system in the supply chain scenario," *IEEE Access*, vol. 7, pp. 115 122–115 133, 2019.

[13] P. Stangl and C. P. Neumann, "FoodFresh: Multi-Chain Design for an Inter-Institutional Food Supply Chain Network," in *Proc of the 14th International Conference on Cloud Computing, GRIDs, and Virtualization (Cloud Computing 2023)*, Nice, France, Jun. 2023, pp. 41–46. DOI: 10.48550/arXiv.2310.19461.

[14] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE International Congress on Big Data (BigData Congress)*, Boston, MA, USA: IEEE, 2017, pp. 557–564. DOI: 10.1109/BigDataCongress.2017.85.

[15] O. Marin, T. Cioara, L. Toderean, D. Mitrea, and I. Anghel, "Review of Blockchain Tokens Creation and Valuation," *Future Internet*, vol. 15, no. 12, p. 382, Nov. 27, 2023, ISSN: 1999-5903. DOI: 10.3390/fi15120382.

[16] E. Takeuchi, "Explaining ethereum contract abi & evm bytecode," Medium, Jul. 16, 2019, [Online]. Available: https://medium.com/@eiki1212/explaining-ethereum-contract-abi-evm-bytecode-6afa6e917c3b (visited on 12/07/2023).

[17] M. M. Chakravarty *et al.*, "The extended utxo model," in *Financial Cryptography and Data Security: FC 2020 International Workshops, AsiaUSEC, CoDeFi, VOTING, and WTSC, Kota Kinabalu, Malaysia, February 14, 2020, Revised Selected Papers 24*, Springer, 2020, pp. 525–539.

[18] Y. Zhou *et al.*, "Stop pulling my rug: Exposing rug pull risks in crypto token to investors," 2024.

[19] H. Huang, W. Kong, S. Zhou, Z. Zheng, and S. Guo, "A survey of state-of-the-art on blockchains: Theories, modelings, and tools," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–42, 2021.

[20] Y. Elmougy and L. Liu, "Demystifying fraudulent transactions and illicit nodes in the bitcoin network for financial forensics," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '23, Long Beach, CA, USA: Association for Computing Machinery, 2023, pp. 3979–3990. DOI: 10.1145/3580305.3599803.

[21] L. Chen *et al.*, "Phishing scams detection in ethereum transaction network," *ACM Trans. Internet Technol.*, vol. 21, no. 1, Dec. 2020, ISSN: 1533-5399. DOI: 10.1145/3398071.

[22] P. Stangl, "Design and Implementation of an Incremental Knowledge Graph Construction Pipeline for Investigating Crypto Asset Fraud," Masterarbeit, Ostbayerische Technische Hochschule Amberg-Weiden, Apr. 2024. DOI: 10.5281/zenodo.14518573.

[23] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 50–70, 2020.

[24] J. Pfeffer, "Ethon: Ethereum ontology," ConsenSys Software Inc., Dec. 7, 2023, [Online]. Available: https://ethon.consensys.io/ (visited on 12/07/2023).

[25] Etherscan, "Etherscan: The ethereum blockchain explorer," Etherscan LLC, Dec. 7, 2023, [Online]. Available: https://etherscan.io/ (visited on 12/07/2023).

[26] Sourcify, "Sourcify: Source-verified smart contracts for transparency and better ux in web3," 2023, [Online]. Available: https://sourcify.dev/ (visited on 12/07/2023).

[27] F. A. Manzano and J. Little, "Pyevmasm: Ethereum virtual machine disassembler and assembler," Crytic, 2024, [Online]. Available: https://github.com/crytic/pyevmasm (visited on 01/25/2024).

[28] F. Victor, "Address Clustering Heuristics for Ethereum," in *Financial Cryptography and Data Security*, J. Bonneau and N. Heninger, Eds., vol. 12059, Cham: Springer International Publishing, 2020, pp. 617–633, ISBN: 978-3-030-51279-8. DOI: 10.1007/978-3-030-51280-4_33.

[29] X Corp., "Filtered stream introduction," X Corp., 2024, [Online]. Available: https://developer.twitter.com/en/docs/twitter-api/tweets/filtered-stream/introduction (visited on 01/25/2024).

[30] M. Hofer, D. Obraczka, A. Saeedi, H. Köpcke, and E. Rahm, "Construction of Knowledge Graphs: State and Challenges," 2023, eprint: 2302.11509 (cs.AI).

[31] Golden Recursion Inc., "Golden Enrichment API: Enrich research, sales, and marketing with fresh, canonical knowledge.," Golden Recursion Inc., 2024, [Online]. Available: https://golden.com/product/api (visited on 01/25/2024).

[32] X. L. Dong *et al.*, "From data fusion to knowledge fusion," *Proc. VLDB Endow.*, vol. 7, no. 10, pp. 881–892, Jun. 2014, ISSN: 2150-8097. DOI: 10.14778/2732951.2732962.

[33] J. Bleiholder and F. Naumann, "Data fusion," *ACM Computing Surveys*, vol. 41, no. 1, pp. 1–41, Jan. 15, 2009, ISSN: 0360-0300, 1557-7341. DOI: 10.1145/1456650.1456651.