The Kosmosis Approach to Crypto Rug Pull Detection

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*—Crypto 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 detect crypto asset fraud are based on the analysis of transaction graphs within blockchain networks. While effective for identifying transaction patterns indicative of fraud, existing approaches do not capture the semantics of transactions and are constrained to blockchain data. Consequently, preventive methods based on transaction graphs are inherently limited. In response to these limitations, we propose the Kosmosis approach, which aims to incrementally construct a knowledge graph as new blockchain and social media data become available. During construction, it aims to extract the semantics of transactions and connects blockchain addresses to their real-world entities by fusing blockchain and social media data in a knowledge graph. This enables novel preventive methods against rug pulls as a form of crypto asset fraud. To demonstrate the effectiveness and practical applicability of the Kosmosis approach, we examine a series of real-world rug pulls. Through this case, we illustrate how Kosmosis can aid in identifying such fraudulent activities by leveraging the insights from the constructed knowledge graph.**

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

I. Introduction

This article is a revised and extended version of [1] and [2]. Crypto assets are digital assets that use distributed ledger technology, such as blockchain, to prove ownership and maintain a decentralized and public ledger of all transactions. Cryptocurrencies, like Bitcoin [3], function as digital currencies and are used for storing or transferring monetary value. Fungible Tokens (FTs), another type of crypto asset, are interchangeable units representing various utilities or assets within a blockchain ecosystem. Lastly, Non-Fungible Tokens (NFTs) are unique digital assets that prove ownership and authenticity of digital or real-world assets [4]. In the rapidly evolving landscape of crypto assets, the incidence of illicit activities has surged. Chainalysis, a leading blockchain analytics firm, reported that illicit transaction volume rose for the second consecutive year in 2022, reaching an all-time high of $20.6 billion in illicit activity [5]. Since the rise of Decentralized Finance (DeFi) in 2020, followed by NFTs in 2021, rug pulls have become a major fraud scheme in terms of amount stolen and frequency [6]. Thus, rug pulls pose a significant risk to investors and undermine the integrity of the crypto asset sector.

The predominant approach for identifying patterns indicative of fraudulent activity is the transaction graph analysis within blockchain networks [7]–[9]. However, this approach presents two key challenges. Firstly, the transacting parties are pseudonymous and only their blockchain addresses are publicly known. This means that, although the transactions of a specific address can be tracked, linking that address to a real-world entity can be challenging since this approach is limited to information or patterns observable in blockchain data. Secondly, this approach is only concerned with the following aspects of a transaction: 1) The transferred asset, 2) the quantity, and 3) the sender and receiver. However, the semantics of a transaction, such as what happened in a transaction that caused the assets to get transferred, is not covered. Thereby limiting the depth of analysis that can be conducted on crypto asset movements.

Knowledge Graphs (KGs) [10] are increasingly recognized as a powerful means to integrate fragmented knowledge from heterogeneous data sources into a graph of data to facilitate semantic querying (e.g., [11][12]) and reasoning (e.g., [13]). A KG provides a holistic view for identifying patterns and hidden connections indicative of fraudulent activities in a highly connected dataset [14]. The KG consists of semantically described entities, each with a unique identifier, and relations among those entities using an ontological representation [15][16]. Their open world assumption allows for the continual integration of new data. By leveraging these capabilities, KGs can enhance crypto asset fraud analysis and aid in predicting future fraudulent activities.

The remainder of this article is organized as follows. We first outline the Kosmosis objectives in Section II. Next, we provide a background on the Ethereum blockchain and graph-based blockchain data mining methods in Section III. Subsequently, in Section IV we propose Kosmosis, our incremental KG construction pipeline. In Section V we provide essential background on rug pulls before we demonstrate the effectiveness and practical applicability of the Kosmosis approach for the use case of NFT rug pull detection in Section VI. Finally, we outline future work in Section VII and conclude the article with a discussion of our findings in Section VIII.

II. Overarching Objectives of Kosmosis

This section outlines the objectives of Kosmosis, beginning with the primary objective that investigates the potential of a KG in identifying and alerting users before they interact with projects linked to known scammers, addressing a critical need for security and trust in blockchain ecosystems. Following that, we explore the technical implications.

Objective 1: How can the KG identify and aid in alerting users before interacting with a rug pull project?

With the rise in illicit activities in the crypto asset market, especially rug pulls, there is a pressing need for effective means to detect and prevent fraudulent activities. Kosmosis aims to integrate fragmented knowledge from blockchains like Ethereum, social media like 𝕏, and potentially other knowledge graphs into one unified KG, enabling semantic querying and reasoning over a graph of entities and the relationships among them. The KG could serve as a knowledge base for a real-time alerting system, warning users of potential risks associated with certain projects or individuals.

Objective 2: How to incrementally construct the KG from heterogeneous data sources?

It is imperative to establish a pipeline capable of integrating updates into the KG in both batch- and streaming-like manner. Thereby, maintaining high data freshness by ensuring that the KG consistently reflects the most up-to-date information from the blockchain and other sources. This approach should not entail a complete reconstruction of the KG, but rather concentrate on integrating new information, avoiding the reprocessing of data that is already incorporated.

Objective 3: How to extract the semantics of blockchain transactions?

Transaction graphs commonly only display transactions with asset transfers and provide answers to questions such as "what" assets were transferred, and "where" were they transferred to. Understanding transactions semantically is vital in uncovering sophisticated fraudulent schemes that might otherwise go unnoticed. Kosmosis addresses this gap by extracting the semantics of transactions, providing answers to "why" and "how" assets were transferred in a transaction. This extraction of semantic information is primarily achieved through decoding the input data of a transaction using the Application Binary Interface (ABI) of smart contracts a transaction interacts with.

III. Background

In this section we provide background on blockchain technology, specifically the Ethereum blockchain, in Section III-A. Subsequently, we outline related graph-based blockchain data mining methods in Section III-B. On the social media aspects of Kosmosis, our prior work includes correlating Reddit data with traditional stock market trends [18] and analyzing Twitter/𝕏 data using SPARQL [19].

A. The Ethereum Blockchain

Blockchain technology is based on the principles of immutability, decentralization, transparency, and cryptographic security and has seen various applications in recent years. For instance, in the financial sector (e. g., [3][20]), or supply chain management (e. g., using a single blockchain [21], or using multiple, interoperable blockchains [22][23]). Smart contract platforms represent a subset of blockchains that enable the development of decentralized applications through smart contracts. This section outlines the key concepts of Ethereum, as an example for smart contract platforms, that are essential for the following sections of this work, such as smart contracts, their execution environment, and account-based accounting.

1) Blockchain Data Structure: A blockchain is a data structure whose elements called blocks are linked together to form a chain of blocks [17], depicted in Figure 1. 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 contracts, 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 [25]. 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

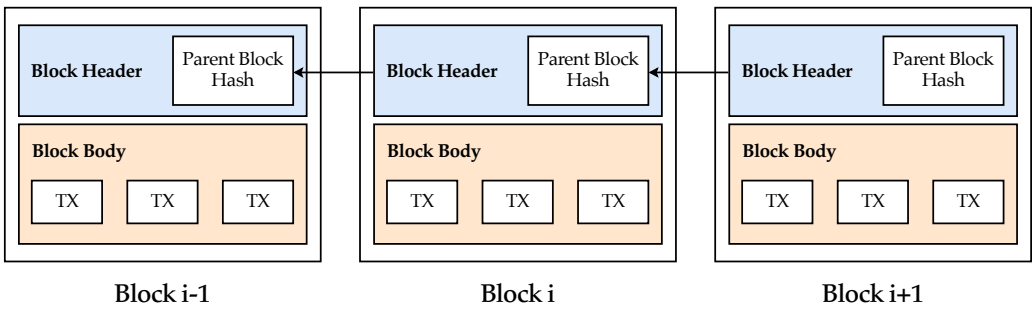

Figure 1. Schematic representation of the blockchain data structure. Adapted from Zheng *et al.* [17].

can create and manage tokens that adhere to the standards ERC-20 for FTs [26] or ERC-721 for NFTs [27]. 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 Ether (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 2 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 entities 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 [3] blockchain that uses the Unspent Transaction Output (UTXO) model, or its successor the extended UTXO [28] utilized by the Cardano [29] blockchain, Ethereum [20] 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: The process of creating new tokens is called token minting. FTs are typically minted by the creator either at the inception of the project or progressively over time. This process is often governed by predefined rules or algorithms embedded within the smart contracts of the project.

In contrast, NFT minting involves other individuals besides the token creator, commonly termed as token minters. They engage by invoking a specific function within a smart contract, in the ERC-721 token standard, called mint. This action results in an increase in the supply of the NFTs and simultaneously assigns these minted tokens to the blockchain address of the minter. The mechanism of minting NFTs often involves utilizing a dedicated minting website. Here, prospective minters or investors are required to invest a predetermined amount, as set by the creator, to initiate the minting process. This investment grants them the ability to mint one or multiple NFTs, depending on the terms set forth in the smart contract.

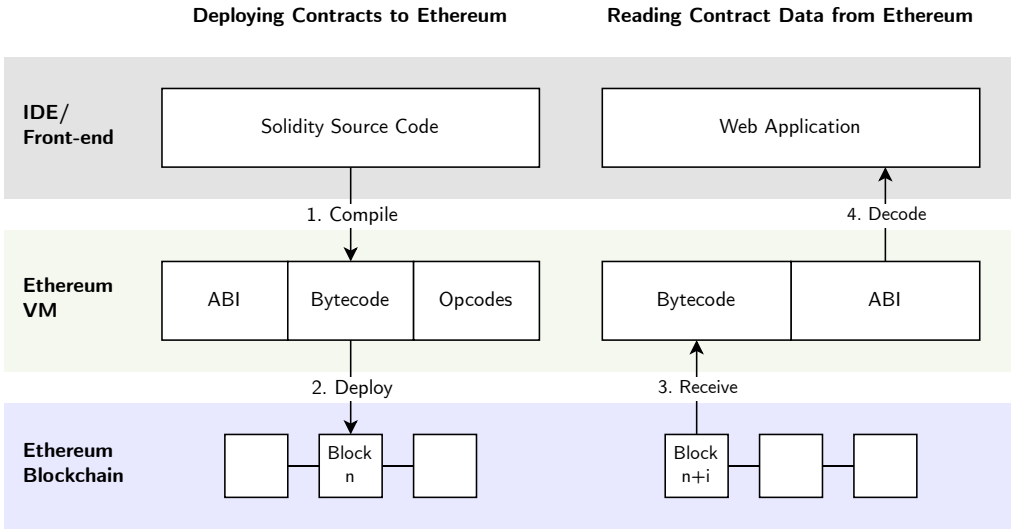

Figure 2. Schematic representation of deploying and reading from smart contracts. Adapted from Takeuchi [24].

B. Rug Pull Detection Methods

Two primary methods have been employed in the past to detect rug pulls: smart contract code analysis and graph-based methods. Smart contract code analysis involves a thorough examination of the contract's code to extract and analyze the semantic behavior of transactions. For instance, [30] utilizes smart contract code analysis to reveal potential vulnerabilities and fraudulent patterns within the contracts. By dissecting the code, their proposed method, dubbed "Tokeer", can identify suspicious patterns and functions that might indicate a predisposition to rug pull scams. Another prominent line of research in smart contract analysis leverages machine learning–based techniques. Mazorra *et al.* [42], for example, employed the XGBoost algorithm as a primary classifier to predict the probability that a liquidity pool will evolve into a rug pull or scam, achieving an accuracy of up to 99.36% using features derived from token propagation patterns and smart contract heuristics. Their dataset and experimental design are restricted to fungible tokens launched exclusively on Uniswap (Ethereum), and the authors argue that directly transferring these learned models to other blockchains is unlikely to yield results of comparable quality. Graph theories and graph-based data mining methods, are applicable for discovering information in blockchain network graphs, because blockchain transactions can be inherently structured into graphs [9]. Elmougy and Liu [31] identified three types of graphs, applicable to any blockchain network: *money flow transaction graphs* visualize the asset flow over time, *address-transaction graphs* showing flow of an asset across transactions and addresses, and *user entity graphs* that clusters the graph for potential linking of addresses controlled by the same user, to deanonymize their identity and purpose. To detect rug pulls with high accuracy, graph-based approaches use network embedding methods to automatically extract features from the blockchain network (e. g., [32]) using a *graph convolutional 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 3. It originated from a master's thesis [33] 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 [34], 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 [35] 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 [36] and Sourcify [37] 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 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* [38]) 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,

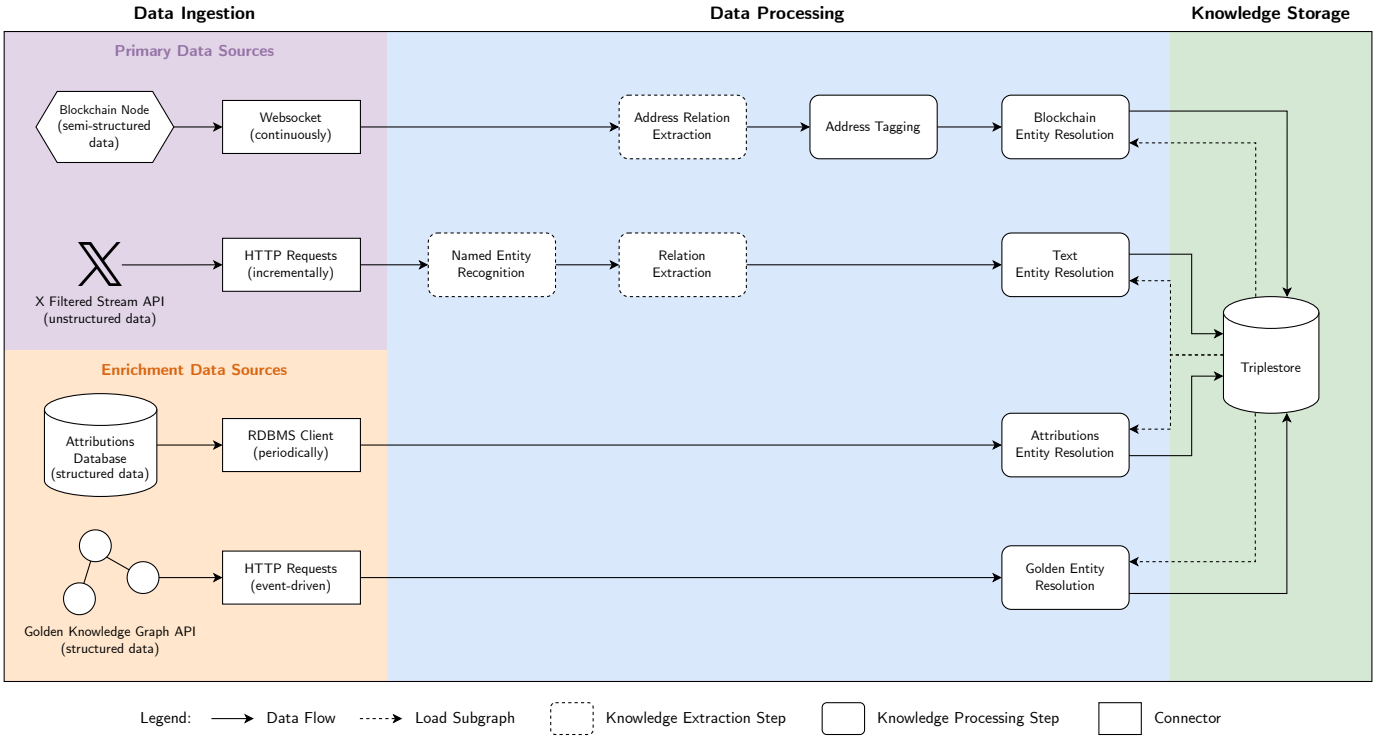

Figure 3. A high level overview of the Kosmosis pipeline.

reflecting the fundamental actions a smart contract on the EVM can perform [20].

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 [39]. 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 [40], 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 [16]. 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 entities. 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 [41]. 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. Rug Pulls and Serial Fraudsters

A rug pull can be categorized as a scam, i. e., the victim authorizes the transaction. This type of scam is typically carried out in five stages, according to [6]: (1) Project creation with

roadmap and total supply of tokens (optional), (2) pre-mint hype, (3) set token mint price, (4) token mint, accumulation of more capital and increase in popularity, and finally (5) the creators cash out, abandon the project, and leave the investors defrauded. To attract users and investments for rug pulls, Sharma *et al.* [6] suggest the involvement of individuals or groups that possess substantial technical skills and knowledge of blockchain technology and demonstrate a proficiency in marketing techniques. This specific use case is particularly relevant given the findings in [6] and [42]: Mazorra *et al.*, who analyzed ERC-20 tokens listed on decentralized exchanges in their 2022 study, labeled 97.7% out of 27,588 analyzed tokens as rug pulls [42]. Likewise, Sharma *et al.* analyzed NFTs and identified a cluster of 168 NFTs associated with what they termed the "Rug-Pull Mafia," a group of creators responsible for orchestrating multiple and repeated rug pulls [6]. There is a growing trend in both the frequency and the financial impact of crypto rug pulls and scams [43]. Notably, the year 2021 marks a peak in the amount stolen, while 2022 shows a sharp rise in the frequency of these fraudulent activities and remains elevated since.

VI. The Use Case of Rug Pull Prevention

To illustrate the vision of Kosmosis-enabled rug pull prevention methods, this section introduces a hypothetical user story centered around a character we name Bob, a crypto market participant. The story telling method of use case illustration was adopted from our previous work in [44, pp. 207–209]. The Kosmosis user story is designed to provide a relatable perspective on how individuals like Bob are affected by such fraudulent activities. The fictional story of Bob is grounded in a series of real-world rug pulls that took place in 2021. All rug pulls were carried out by the same fraudulent NFT creator and Twitter user known as Homer_eth. In Section VI-C, we outline how the series of rug pulls experienced by Bob might have unfolded differently had he been equipped with a Kosmosis-enabled fraud prevention mechanism at the time.

A. Past User Story

In the span of two months, from October to November 2021, a fraudulent NFT creator and 𝕏 user known as Homer_eth executed five different NFT project rug pulls within two months, accumulating over $2.8 million in profits. Table I provides an overview of Homer_eth's rug pull projects, each with launch date and the estimated profit.

TABLE I. Rug Pull Projects by Homer_eth

Project Name	Launch Date	Estimated Profit
Ether Bananas	10/07/2021	$125k
Ether Monkeys	10/11/2021	$1.77m
Zombie Monkeys	10/15/2021	$413k
Ether Reapers	10/20/2021	$282k
ETH Banana Chips	11/23/2021	$208k

The basis of this user story is the transaction graph depicted in Fig. 4 that provides a simplified visualization of the transaction flow across multiple NFT projects linked to Homer_eth. It highlights key components, including EOA Nodes (Externally Owned Accounts), which represent the multiple wallet addresses of the rug puller, and Deployer Nodes (Smart Contract Creators), with the 0xc8a6 address being the deployer for multiple fraudulent contracts. The links between addresses are established through various transaction, such as mint transactions (e.g., mintReaper, mintBananaChips), which indicate purchases; fund transfers (e.g., Transfer 65.61 ETH to 0xc8a6), showing proceeds flowing to personal wallets or exchanges; and contract deployments (e.g., Deploy Ether Reapers). The transaction graph makes a critical indication of fraudulent intent visible. Instead of using a multisig treasury or project contract, funds were immediately funneled to a single address controlled by the rug puller.

Bob's story begins with a common enthusiasm for the burgeoning world of NFTs. His journey into the NFT market is marked by excitement and optimism, spurred by the success stories he sees online. Homer_eth, an NFT creator and 𝕏 user, has caught the attention of many like Bob by sharing his NFT

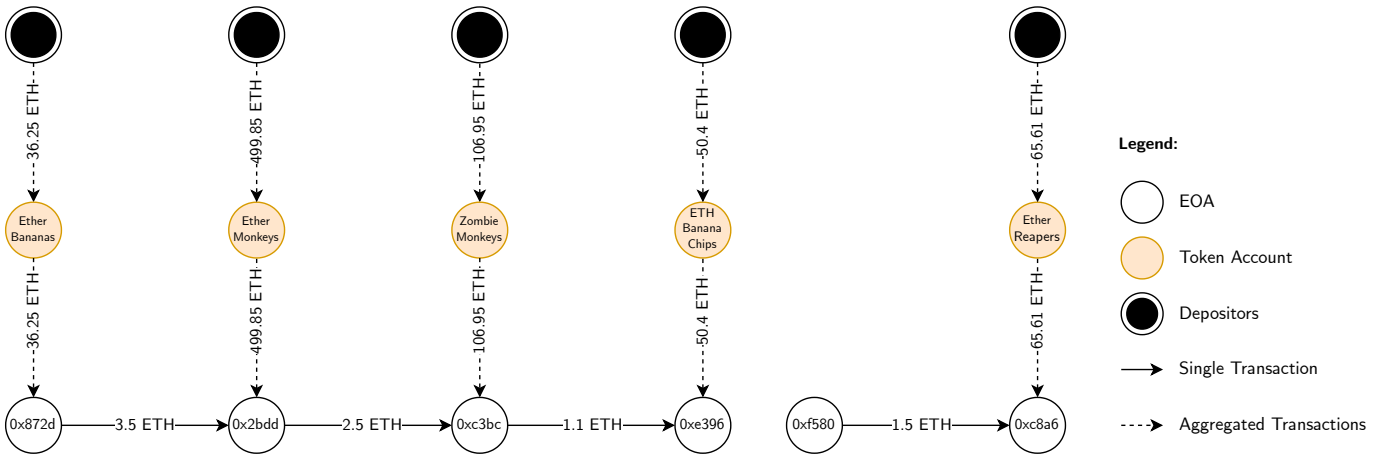

Figure 4. Simplified transaction graph of Homer_eth's NFT rug pulls.

projects on 𝕏. His first NFT collection was *Ether Bananas*, consisting of 750 NFTs, was launched on October 7, 2021. Only four days later, on October 11, Homer_eth continued with the release of *Ether Monkeys*, followed by the release of *Zombie Monkeys*. The buzz around Homer_eth's projects, especially *Ether Monkeys*, which promised additional utility through a casino to gamble and a decentralized autonomous organization to govern the NFTs, according to [45], draws Bob into the fray. Being relatively new to the NFT market, Bob views this as an opportunity not to be missed. Bob bought his first NFT from Homer_eth, an *Ether Reapers*, and with that purchase, he was no longer just a bystander; he was now 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 reveled in the rumors of more NFT launches in the future. His commitment paid off when he earned himself a whitelist spot that allows Bob to mint the upcoming NFT project *ETH Banana Chips* by Homer_eth. Convinced of its potential, Bob didn't hesitate to mint an *ETH Banana Chips* NFT when the opportunity arose. With a click to confirm the transaction in his browser wallet (e. g., MetaMask [46]), Bob became the proud owner of an *ETH Banana Chips* NFT, unaware of the underlying risks associated with his investment.

However, the reality of the situation was far from the optimistic scenario Bob had envisioned. Unknown to him, since Bob had a limited understanding of blockchain transactions, the proceeds from the *Ether Reapers* mint were not being locked in the smart contract for future development as promised. Instead, they were directly funneled into Homer_eth's deployer address. From there, Homer_eth will later transfer those mint proceeds either to his next deployer address for a future rug pull or to an exchange in order to pay out his profits made from rug pulling the projects.

After the launch of *ETH Banana Chips*, a tense silence enveloped the community. For months, there was no news from Homer_eth, no updates on the project, leaving everyone to wonder about the future. It wasn't until March 2022, that Homer_eth broke the silence with the announcement of one last NFT project, dubbed *Froggy Frens*. However, due to backlash from the community, Homer_eth deleted his 𝕏 account and vanished [45].

B. Kosmosis Extension

The basis of the extended user story is the Kosmosis KG, depicted in Figure 5. 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

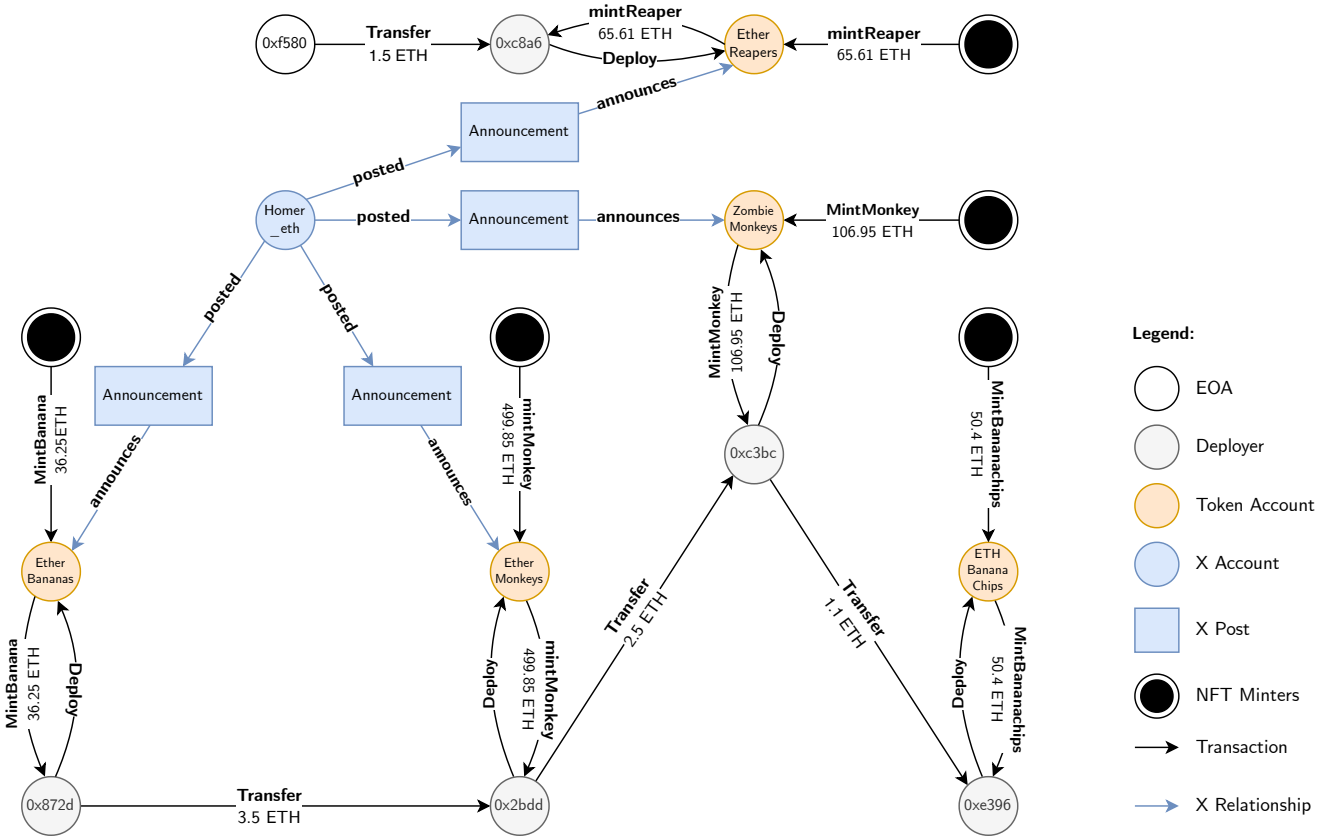

Figure 5. Knowledge graph of Homer_eth's NFT rug pulls, constructed using Kosmosis.

the primary method of associating blockchain transactions with social identities, which enhances the semantic richness of the constructed KG. The Kosmosis KG for this specific user story in Figure 5 is a direct enhancement over the basic transaction graph from Figure 4 as it was discussed in the previous subsection. The enhancement comprises semantically annotated edges and the incorporation of data from the social media platform 𝕏.

In order to detect potentially suspicious activity, we construct a Kosmosis subgraph using a multi-part SPARQL query. Listing 1 provides the overall CONSTRUCT query statement, capturing Ethereum transfers, mint transactions, smart contract deployments, bridging activities between chains, and social media references to blockchain addresses. Optional name labels help to identify the connection between blockchain or social media accounts and their associated real-world entities.

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
PREFIX kos:  <http://oth-aw.de/kosmosis#>

CONSTRUCT {
  # transfers
  ?sender    kos:transferTo ?receiver .
  ?trEdge  a kos:TransferEdge ;
             kos:from  ?sender ;
             kos:to    ?receiver ;
             kos:value ?sumValueETH .

  # contract deployments
  ?deployer  kos:deployed ?contract .

  # mint flows
  ?minter     kos:mint ?nftContract .
  ?mintEdge a kos:MintEdge ;
              kos:from  ?minter ;
              kos:to    ?nftContract ;
              kos:value ?sumMintETH .

  # UTXO unlock
  ?utxoIn      kos:unlocks ?contractUtxo .

  # bridging
  ?bridgeFrom  kos:depositToBridge ?bridge .
  ?bridge      kos:bridgeTransfer  ?bridgeTo .

  # social layer
  ?xAccount    kos:posted   ?xPost .
  ?xPost       kos:announces ?announcedContract .

  ?anyAccount  kos:accountName ?accName .
}
```

Listing 1. Construction of the RDF subgraph.

To enhance the interpretability of raw blockchain transactions, we introduce semantic annotations in our knowledge graph. This process involves using the Application Binary Interface (ABI) of smart contract transactions. Transactions interacting with NFT minting contracts contain function calls embedded in input data. Using the ABI, we extract function names (e.g., mintMonkey, mintBananaChips) and parameters. This enables labeling edges as minting transactions rather than generic asset transfers. Transaction type classification is done by categorizing transfers into value transactions, such as *mintMonkey* and *Transfer*, and non-value transactions, like contract deployments denoted as *Deploy*. These semantics allow describing (i. e., tagging) sender and receiver addresses as NFT minter (previously depositor) and deployer (previously EOA).

As part of the subgraph construction, multiple Ethereum value transactions between EOAs are aggregated in Listing 2. The query calculates the total ETH transferred from each sender to each receiver and assigns a unique semantic identifier to these aggregated transfers. This process reduces transactional complexity while preserving critical relationships for identifying significant value flows indicative of suspicious activity.

```
WHERE {
  {
    SELECT ?sender ?receiver (SUM(?value) AS ?sumValueETH)
    WHERE {
      ?tx a kos:ValueTransaction ;
            kos:from    ?sender ;
            kos:to      ?receiver ;
            kos:value   ?value ;
            kos:minedOn kos:Ethereum .
      ?sender   a kos:ExternallyOwnedAccount .
      ?receiver a kos:ExternallyOwnedAccount .
    }
    GROUP BY ?sender ?receiver
    HAVING (?sumValueETH > 0)
  }
  BIND( IRI(CONCAT("urn:tx-agg:",
        SHA256(CONCAT(STR(?sender),STR(?receiver))))) AS ?trEdge )
```

Listing 2. Aggregated ETH transfers between EOAs on Ethereum.

Due to the categorization of blockchain transactions by their semantic functions further important transaction types can be captured. Specifically, Listing 3 collects direct Ethereum smart contract deployments, aggregates Ethereum NFT mint transactions according to function names embedded in transaction data, and incorporates Bitcoin UTXO transactions. Each aggregated transaction type is semantically annotated to clarify the nature of the underlying blockchain interaction.

Cross-chain interactions through bridge protocols are captured explicitly in Listing 4. Ethereum deposits made into bridge smart contracts and their subsequent transfers to Polygon-based externally owned accounts are identified and annotated. This detailed semantic labeling assists in tracing asset movements across blockchains, crucial for identifying potentially risky bridging behavior.

Finally, the data integrated from platform 𝕏 enriches the KG with detailed information about user accounts, labeled as *X Account*, and specific announcements or posts, referred to as *X Post*. This integration facilitates a deeper understanding of the context and relationships surrounding these rug pulls. For

```
# Direct deployments
UNION
{
  ?deployer bco:deployed ?contract .
}

# Mint transactions
UNION
{
  SELECT ?minter ?nftContract (SUM(?val) AS ?sumMintETH)
  WHERE {
    ?mintTx a kos:CallTransaction ;
              kos:from  ?minter ;
              kos:calls ?nftContract ;
              kos:value ?val ;
              kos:action ?funcName .
    FILTER regex(?funcName, "^mint", "i")
    ?nftContract a kos:TokenAccount .
  }
  GROUP BY ?minter ?nftContract
}
BIND( IRI(CONCAT("urn:mint-agg:",
      SHA256(CONCAT(STR(?minter),STR(?nftContract))))) AS
    ?mintEdge )

# UTXO transactions
UNION
{
  ?utxoIn a kos:TransactionInput ;
            kos:unlocks ?contractUtxo .
}
```

Listing 3. Contract deployments and aggregated 'mint' calls on Ethereum, as well as UTXO transactions on Bitcoin.

instance, the KG can establish a connection between previously unrelated entities, such as the deployer address *0xc8a6* and the user Homer_eth. This connection is made through a social media announcement in which Homer_eth claims to have created the Ether Bananas project, as well as through semantic annotation, which identifies *0xc8a6* as the deployer address of the Ether Bananas smart contract.

The final part of the query in Listing 5 retrieves social media data integrated from platform 𝕏, focusing on posts that explicitly announce (e.g., direct claim of creation: "Proud to announce the launch of Ether Bananas!") or reference blockchain contracts in the comments section (e.g., "The CA is 0xCeF4CCb03dbc7D87B388407e381a949bE6d00E3b," where CA stands for contract address of the NFT). It associates user identities (𝕏 accounts) and their posts with blockchain addresses they mention or promote. Optionally, the query includes usernames or labels from social accounts, further enhancing the contextualization of blockchain entities that allows to directly to real-world social identities, if possible.

```
UNION
{
  ?depTx  a         kos:ValueTransaction ;
          kos:from  ?bridgeFrom ;
          kos:to    ?bridge ;
          kos:value ?depVal .
  ?bridge a kos:ContractAccount .

  ?transTx a         kos:ValueTransaction ;
           kos:from  ?bridge ;
           kos:to    ?bridgeTo ;
           kos:value ?depValPolygon .
  ?bridgeTo a kos:ExternallyOwnedAccount ;
            kos:existsOn kos:Polygon .
}
```

Listing 4. Bridge deposits and transfers from and to the Ethereum blockchain.

```
UNION
{
  ?xAccount a kos:XAccount ;
              kos:posted ?xPost .
  ?xPost    a kos:XPost ;
              kos:announces ?announcedContract .
}
OPTIONAL { ?anyAccount bco:accountName ?accName }
}
```

Listing 5. 𝕏 Posts that announce contracts. Optionally, carrying over any account name literals, if present.

In conclusion, the Kosmosis KG provides the semantic data foundation necessary for sophisticated detection logic. By aggregating transactions and enriching them with semantic annotations, the system can detect suspicious patterns, such as rapid bulk transfers following token minting events. Such transactions can be assigned elevated risk scores based on correlated indicators (e.g., rapid withdrawal to external accounts controlled by the deployer), triggering timely automated alerts and significantly reducing the reaction time required to prevent potential rug pulls.

C. Future User Story

In an alternative scenario where Bob would have used Kosmosis, it would have analyzed the transaction history. The system would have issued a rug pull warning based on patterns of fund diversion to deployer addresses. Bob's journey in the NFT market would have been safer, beginning with his initial transaction to purchase an *Ether Reapers* NFT.

As soon as Bob initiated his transaction, the rug pull prevention mechanism would have accessed the KG, to analyze the rug pull risk of the contract. Based on the integrated knowledge from 𝕏, the system would have been able to link the contract, Bob is about to interact with, to all of Homer_eth's prior blockchain activity. The KG would have revealed a critical anomaly. Instead of the mint proceeds being transferred to the contract address of the project for future development, they

were being diverted to the *Ether Reapers* deployer address via the *MintReaper* function. With smart contracts acting as an automated and trustless intermediary, where the code of the contract dictates the flow of funds according to predefined rules, this pattern of fund diversion is absent in legitimate projects. When funds are sent directly to a team member's address, in this case the deployer address, the funds can be moved to exchanges or other addresses with ease (i. e., pulling liquidity from the project without fulfilling the promises). This is a common tactic in rug pulls, where the developers abandon the project and disappear with the investor funds. Therefore, signaling a potential rug pull behavior. Upon detecting this anomaly, the system would have immediately issued a rug pull warning to Bob, prompting Bob to make an informed decision by asking whether he wishes to proceed with the transaction despite the identified risk. This proactive approach empowers Bob to reconsider his decision with full awareness of the potential danger, offering him a chance to opt-out before potentially falling victim to a rug pull.

VII. Current Limitations

We aim to translate the Kosmosis approach into a practical implementation. 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.

The generalization, from the exemplary use case to a sophisticated general rug pull classification method, covering various data patterns in the KG, is open research. Our subsequent endeavor involves the development of an algorithm capable of discerning rug pull warnings at varying confidence levels. This pursuit commences with the formulation of an intricate SPARQL query. Furthermore, an alerting system that utilizes the KG, constructed with Kosmosis, to alert users before interacting with a potential rug pull project, as described in the user story of Section VI, requires future efforts.

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 [47], 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 [48], knowledge fusion would allow for a more accurate representation of real-world entities 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.

VIII. Conclusion and Future Work

The Kosmosis approach represents a significant advancement in addressing the challenges associated with crypto rug pulls. Our proposed approach offers enhanced capabilities for semantic analysis, allowing the identification of fraud patterns that traditional transaction graph methods cannot detect.

We outlined a user story, where a threat actor known as Homer_eth executed five NFT project heists within two months, accumulating over $2.8 million in profits. In such a scenario, we showed that Kosmosis provides a knowledge graph that improves the detection of such fraudulent schemes carried out through sophisticated transaction patterns that might otherwise go unnoticed in related approaches, such as smart contract code analysis. This capability helps users make informed decisions and avoid becoming victims of fraud.

We also demonstrated how to build a knowledge graph from blockchain and social media data using the Kosmosis approach to incremental knowledge graph construction. Kosmosis 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.

Kosmosis pipeline supports the ingestion of unstructured, semi-structured, and structured data, as well as the ingestion of new data at different time intervals. During construction, the semantics of blockchain transactions are extracted to address "why" and "how" crypto assets were transferred. Thus, Kosmosis extends the traditional transaction graph into a *semantically enhanced transaction graph* in which the sender and recipient are still pseudonyms. By incrementally constructing a *knowledge graph from blockchain and social media data*, Kosmosis also bridges the gap between pseudonymous transactions and real-world entities.

References

[1] P. Stangl and C. P. Neumann, "Kosmosis: Crypto Rug Pull Detection and Prevention by Fusing On- and Off-Chain Data in a Knowledge Graph," in *Proc of the 16th International Conference on Cloud Computing, GRIDs, and Virtualization (Cloud Computing 2025)*, Valencia, Spain, Apr. 2025, pp. 1–8. DOI: 10.5281/zenodo.17272133.

[2] 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 2025)*, Bamberg, DE, Mar. 2025. DOI: 10.18420/BTW2025-131.

[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008, [Online]. Available: https://bitcoin.org/bitcoin.pdf (visited on 12/17/2023).

[4] S. Alizadeh, A. Setayesh, A. Mohamadpour, and B. Bahrak, "A network analysis of the non-fungible token (nft) market: Structural characteristics, evolution, and interactions," *Applied Network Science*, vol. 8, no. 1, p. 38, 2023.

[5] Chainalysis Inc., "The 2023 crypto crime report," Chainalysis, Feb. 2023, [Online]. Available: https://go.chainalysis.com/2023-crypto-crime-report.html (visited on 12/17/2023).

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] A. Hogan *et al.*, "Knowledge Graphs," *ACM Computing Surveys*, vol. 54, no. 4, pp. 1–37, May 31, 2022, ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3447772. arXiv: 2003.02320 [cs.AI]. [Online]. Available: http://arxiv.org/abs/2003.02320 (visited on 12/11/2023).

[11] A. Chernysheva *et al.*, "SGDb Semantic Video Game Database: Svelte- und Ontotext-basierte Webanwendung mit einer Graphen-Suche für Videospiele," German, Ostbayerische Technische Hochschule Amberg-Weiden, Technische Berichte CL-2023-02, Mar. 2023. DOI: 10.13140/RG.2.2.11272.60160.

[12] J. Halbritter *et al.*, "Graphvio: Eine Graphdatenbank-Webanwendung für integrierte Datensätze von Streaminganbietern," German, Ostbayerische Technische Hochschule Amberg-Weiden, Technische Berichte CL-2022-01, Mar. 2022. DOI: 10.13140/RG.2.2.12111.46244.

[13] C. P. Neumann, T. Fischer, and R. Lenz, "OXDBS – Extension of a native XML Database System with Validation by Consistency Checking of OWL-DL Ontologies," in *Proc of the 14th International Database Engineering & Applications Symposium (IDEAS'10)*, Montreal, QC, CA, Aug. 2010, pp. 143–148. DOI: 10.1145/1866480.1866502.

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

[15] C. Feilmayr and W. Wöß, "An analysis of ontologies and their success factors for application to business," *Data & Knowledge Engineering*, vol. 101, pp. 1–23, 2016.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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).

[21] 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.

[22] 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.

[23] P. Stangl, "Design and Implementation of a Heterogeneous Blockchain Consortium for a Food Supply Chain Network," Bachelor's Thesis, Ostbayerische Technische Hochschule Amberg-Weiden, Jan. 2022. [Online]. Available: https://www.cyberlytics.eu/theses/all/OTH-AW/BT_2022_Stangl_Philipp_Thesis/BT_2022_Stangl_Philipp_Thesis.pdf.

[24] 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).

[25] 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. (visited on 12/17/2023).

[26] F. Vogelsteller and V. Buterin, "ERC-20: Token Standard," Ethereum, Nov. 19, 2015, [Online]. Available: https://eips.ethereum.org/EIPS/eip-20 (visited on 12/17/2023).

[27] W. Entriken, D. Shirley, J. Evans, and N. Sachs, "ERC-721: Non-Fungible Token Standard," Ethereum, Jan. 24, 2018, [Online]. Available: https://eips.ethereum.org/EIPS/eip-721 (visited on 12/16/2023).

[28] 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.

[29] C. Hoskinson, "Why we are building Cardano," IOHK, 2017, [Online]. Available: https://api-new.whitepaper.io/documents/pdf?id=HkUIhFWhL (visited on 12/16/2023).

[30] Y. Zhou *et al.*, "Stop pulling my rug: Exposing rug pull risks in crypto token to investors," in *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*, ser. ICSE-SEIP '24, Lisbon, Portugal: ACM, 2024, pp. 228–239. DOI: 10.1145/3639477.3639722.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

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

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

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

[38] 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).

[39] 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. (visited on 12/03/2023).

[40] 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).

[41] 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).

[42] B. Mazorra, V. Adan, and V. Daza, "Do not rug on me: Zero-dimensional scam detection," 2022, eprint: 2201.07220 (cs.CR).

[43] R. Moody, "Worldwide crypto & nft rug pulls and scams tracker," Comparitech, Nov. 2023, [Online]. Available: https://www.comparitech.com/crypto/cryptocurrency-scams/ (visited on 11/19/2023).

[44] C. P. Neumann, *Distributed Case Handling*. München: Verlag Dr. Hut, 2013, ISBN: 9783843909198.

[45] 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).

[46] MetaMask, "Metamask: A crypto wallet & gateway to blockchain apps," ConsenSys Software Inc., 2023, [Online]. Available: https://metamask.io/ (visited on 12/15/2023).

[47] 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.

[48] 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.