Rule-based Blockchain Knowledge Graphs: Declarative AI for Solving Industrial Blockchain Challenges

Luigi Bellomarini[1], Giuseppe Galano[1], Markus Nissl[2], and Emanuel Sallinger[2,3]

[1] Banca d'Italia, Italy
[2] TU Wien, Austria
[3] University of Oxford, UK

Abstract. Blockchains have an incredible impact on our economic systems. Over the last years, more and more illicit activities have taken place not only via the traditional financial space but also via cryptocurrencies, where tracking illicit activities is more complex. Yet, several blockchain toolkits have been introduced that explore blockchain data. There is a clear industrial need to bring together many of those valuable methods into a principled approach that upholds central aspects such as the ability to provide explainable analytics.
In this paper, we report on an international industrial and academic collaboration that uses a declarative, rule-based approach on top of a Knowledge Graph system and describes the use-case of analyzing the Petya ransomware.

Keywords: Blockchain Analytics · Knowledge Graphs.

1 Introduction

Business Case and Importance. Blockchains have an incredible impact on our economic systems with a total market capitalization of 1500 billion US Dollar [8]. Over the last years, a high number of financial transactions has been executed via blockchains, leading to increased interest in different areas, such as analytics or fraud detection. Detecting fraud and flagging illicit activities is one of the key areas in the financial system. This requires the ability to detect patterns, to *reason* over both the traditional financial space as well as via cryptocurrencies. Thus, it is of primary importance for financial institutions in the European and world economic systems to reason over blockchain data.

Technological Challenges. While several blockchain toolkits have been introduced that include different algorithms for several tasks [3, 9, 12], there is a

The views and opinions expressed in this paper are those of the authors and do not necessarily reflect the official policy or position of Banca d'Italia.

clear industrial need to bring together many of those valuable, but ad-hoc combined methods into a principled approach. This principled approach must uphold central aspects such as the ability to provide *explainable analytics*, so that the derived insights are based on a solid foundation. Meeting these challenges is one of the traditional strong points of rule-based, declarative AI.

Solution. Researchers have recognized [9, 22] that blockchains share common features with graph-like data structures and started modelling blockchain data as a direct acyclic graph. This is possible thanks to the structural guarantees of the rules that govern blockchain operation, such as that one block references only one predecessor or that each block contains a logically ordered set of transactions. By interpreting the blockchain as a graph, special graph traversal algorithms can be used for querying the data such as calculating the shortest path between two nodes, e.g., accounts, or generating sub-graphs [9].

In this paper, we introduce a connection on top of the graph by interpreting the blockchain as a logic-based Knowledge Graph. We show that blockchains share different research questions with the Knowledge Graph domain. One particular area of such relationship is given by detecting illicit activities such as fraud or ransomware payments [17], which include several tasks such as identifying the real-world entities behind users [20], clustering of blockchain identities [24] and devising economic indicators [22].

Apart from this general connection, this paper is motivated by the challenge: *can a rule-based, declarative Knowledge Graph be used to provide an end-to-end analysis of the Petya ransomware*, a particularly widely discussed case in media, e.g., [1, 23]. More concretely, we use the Knowledge Graph system Vadalog [4] that is able to handle heterogeneous data sources and allows to formulate declarative, and full-recursive queries extending Datalog. By integrating existing analytical frameworks in the Knowledge Graph, we are able to formulate a wide range of use-cases in a rule-based format over the blockchain as well as the entire analytical frameworks. This gives us the possibility to provide answers to central questions in the financial domain within one coherent system.

Main Contributions. To summarize, our main contributions are:

- We identify a novel relationship between blockchains and Knowledge Graphs, which goes beyond the graph-like structure. We show that reasoning in Knowledge Graphs share common tasks with typical blockchain workflows.
- We present a rule-based approach for detecting illicit activities in blockchains. By using Datalog rules we provide a transparent and declarative way to formalize specific questions.
- We show our approach by constructing and querying the Petya ransomware graph. We provide several non-trivial answers to industrial use-cases.

Organization. The remainder of this paper is organized as follows: In Section 2 we provide background information required in the following chapters. Section 3 explains the relationship between a Knowledge Graph system and blockchains in detail. In Section 4 we provide an in-depth analysis of the Petya ransomware. We provide related work in Section 5 and conclude the paper in Section 6.

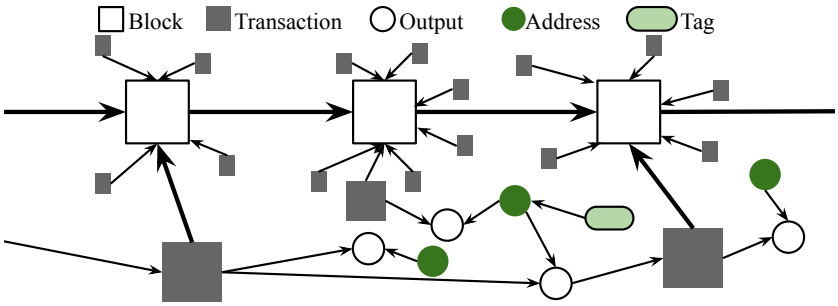

Fig. 1. Graph view on Bitcoin structure.

2 Preliminaries

Blockchains. A blockchain is a type of distributed ledger, where blocks are structured in a single-linked list which is linked with a hash reference to the previous block. Each block contains a list of transactions, which contain information required to transfer coins (and data) between different accounts. An account is created by a private and public key pair. The private key is used to spend the coins, while the (hash of) the public key is used to receive the coins, hence called the address of the account. In Bitcoin, each transaction can have multiple inputs and outputs either from one or from different users. An input is used to spend a previous output, which creates an acyclic graph of spent outputs. The difference between the inputs (i.e., output values of the previous transaction) and the output values in one transaction are the fee paid to the miner of the block [19]. Figure 1 gives a visualization of the Bitcoin structure as a graph, summarizing the discussed concepts. White square represents blocks, dark squares stand for transactions, white circles are outputs (and inputs), and dark (green) circles represent addresses. In addition, the figure contains tags of addresses, which are an external information required for blockchain analytics.

Mixing services. Mixing services are used to enhance the privacy of the user. While mixing services are embedded into the protocol in some blockchains, the usage in Bitcoin requires expertise. In case of Bitcoin, mixing tries to hide the mappings between the inputs and outputs of the transaction taking the transferred values in consideration to obfuscate which input and output address belongs to the same user. In simple words, a perfectly mixed transaction would have the same value for each input and output. It turned out, that many mixing attempts are not perfect. Address-reuse, same output or input addresses, or different output values make mixing attempts more or less useless [15]. Boltzmann [14] is a script that analyzes the so-called linkability of a single transaction without considering information from different transactions. The linkability between a specific input and output of a transaction is defined as the ratio between the number of valid mappings between inputs and outputs containing the specific input and output and possible valid mappings given the values of the transaction.

Knowledge graphs. The definitions of the term Knowledge Graph are manifold [5]. In this paper, we use the definition of Bellomarini et al. [4], who describe a Knowledge Graph as a *semi-structured data model characterized by three components: (i) a ground extensional component, that is, a set of relational constructs for schema and data (which can be effectively modeled as graphs or generalizations thereof); (ii) an intensional component, that is, a set of inference rules over the constructs of the ground extensional component; (iii) a derived extensional component that can be produced as the result of the application of the inference rules over the ground extensional component (with the so-called "reasoning" process).* We will use this definition in Section 3 to show the fundamental connection between Knowledge Graphs and blockchain analytics.

3 Knowledge Graphs for Blockchains

As discussed in the previous sections, the Bitcoin blockchain structure can be interpreted as a graph-like structure for different blockchain applications. While this connection builds the fundamental relationship to Knowledge Graphs by mapping the Bitcoin transaction structure to the extensional component of the Knowledge Graph, we can identify several additional properties that overlap between blockchains and the Knowledge Graph system. All the mentioned tasks in Section 1, such as clustering, have in common that they infer new knowledge based on existing data, which can be seen as generating new nodes and edges (the derived extensional component) by using the intensional component. Hence, Knowledge Graphs and blockchain tools share additional functionalities:

- **Shared reasoning tasks.** The typical behavior of tasks involving blockchains highly relates to Knowledge Graph reasoning. For example, identifying the real-world entities behind accounts depends on the clustering of several accounts into one real-world entity, which is comparable to entity resolution [7], i.e., decide whether two accounts are the same real-world entity, and link prediction [25], i.e., decide whether the account is part of the real-world entity.
- **Shared interference rules.** Knowledge graph and blockchains share operationalized domain knowledge that is necessary to model domain-specific rules and patterns such as the representation of known money laundering or fraud schemes for analytical tasks.

While so far our focus has been on the abstract connection, we now concentrate on a very concrete use-case, namely the *Petya ransomware challenge*. We do want to mention that the provided description will only focus on the transaction level, yet Knowledge Graphs are also made to perfectly model smart contracts as part of the intensional component, i.e., the contract as rules. Smart contracts are applications living inside the blockchains such as lottery services or supply chain applications.

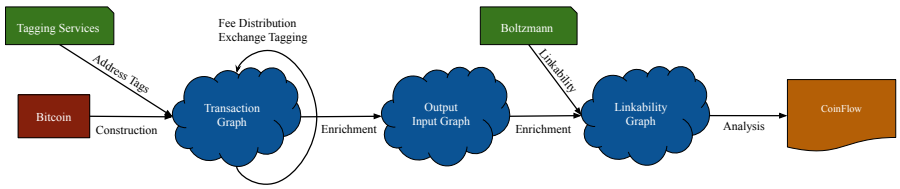

Fig. 2. High-Level Workflow of the Petya Analysis

4 Use Case: Tracking the Petya Ransomware

In this main section, we report how our approach performed on a concrete use-case, namely the analysis of the Petya ransomware. This is only one example, but representative for demonstrating how a concrete use-case can be solved using a rule-based Knowledge Graph system. In fact, it requires to give answers to three questions, spanning the entire life cycle of Knowledge Graphs:

- How to **construct** a Knowledge Graph for analyzing the Petya ransomware in order to include relevant data sources. (Section 4.1)
- How to **discover** new facts based on the data in the Knowledge Graph to get better insights. (Section 4.2)
- How to formulate and use **reasoning** queries to perform blockchain analytics and extract relevant information. (Section 4.3)

In Figure 2 we detailed the concrete steps of the workflow to address the previously described questions. It begins with the construction by using the blockchain and tagging services, then moves on to handling distinct discovery processes (e.g., fee distribution, linkability) on three different graph perspectives, before concluding with a coinflow analysis as a reasoning task. We will provide details on each part in the respective subsections.

4.1 Petya Knowledge Graph Construction

Knowledge Graph construction has the goal of collecting and modeling relevant data sources as the extensional component in a Knowledge Graph system. In the case of the Petya Knowledge Graph, we use two different data sources, namely *(i)* the Bitcoin blockchain itself for the basic data and *(ii)* tags to Bitcoin addresses belonging to exchanges (i.e., services that swap different cryptocurrencies with other parties) from different reliable online sources.

Bitcoin Model. There exist different views on the Bitcoin blockchain, for example the default structure of the blockchain as defined in Section 2, or the linking of transactions on the transaction or on the output level. Since efficient Knowledge Graph Management Systems allow to convert between different views with negligible performance overhead, we define the complete blockchain structure as our extensional component of the KG and then convert this structure to different

views with rules for analysis during the discovery step (see Section 4.2, where we will show graphical illustrations). The main relations are:

$$Block(blockHash, number, prevBlock) \quad (1)$$

$$Tx(txHash, blockHash) \quad (2)$$

$$TxIn(txHash, inIndex, spentTxHash, spentTxOutIndex) \quad (3)$$

$$TxOut(txHash, outIndex, value) \quad (4)$$

Relations 1-4 define the structure of the blockchain. Each transaction is part of a block and has several inputs and outputs. An input references to an output, which it consumes, while a block references to its predecessor (as described in Section 2). We want to point out that any additional attributes, like the address of the output, will be provided by additional relations, if required, e.g., *TxOutAddr*(txHash, outIndex, addr).

Exchange Model. For modelling exchanges we use facts in the Knowledge Graph that mark an address as an exchange, i.e., *Exchange*(addr).

Petya Blockchain Source. For Petya, we extracted the Bitcoin blockchain data only in the relevant time frame of the event, i.e., between the blocks 473.000 and 490.000. In detail, we scanned the blockchain for transactions sending money to the ransomware address "1Mz7153h..." and imported all following transactions in the Knowledge Graph until we reached one of our three termination criteria: (i) We identified an exchange by checking the output address against the exchange model. (ii) We reached the end of our interval of interest, i.e., block number 490.000. (iii) We exceeded a depth-limit of ten in our search graph. This depth-limit was chosen due to the exponentially increased breadth of the transaction tree in the deeper levels and the reduced likelihood that the coins in the transaction are still belonging to the Petya user.

4.2 Petya Knowledge Graph Discovery

Knowledge Graph discovery extends the Knowledge Graph by deriving the extensional component through applying rules of the intensional component. In case of Petya, we extend the Knowledge Graph during discovery with additional facts to gain more insight during the analysis. These facts can be either added by our own rules or by rules interacting with other blockchain analytic frameworks. In order to integrate additional knowledge, it is required to answer the following three highly important questions:

- How can we use different views of the blockchain for analytical tasks?
- How can we use data provided by other analytical tools? We show this by calculating the linkability (introduced in Section 2) between in- and outputs.
- How can we use existing knowledge to extend the Knowledge Graph with our own rules? We show this by extending the list of exchanges.

In order to answer these questions, we show rules that describe and highlight the logic-based approach. These rules are given and executed in Vadalog [4].

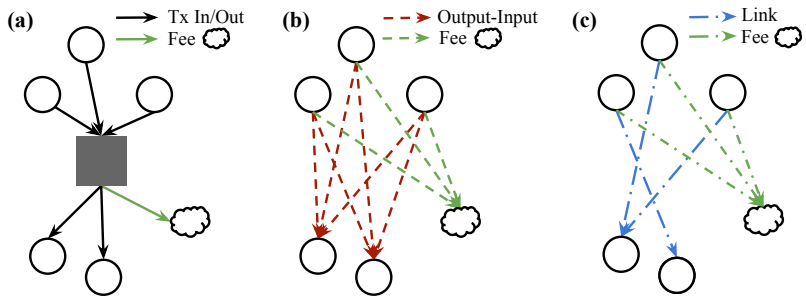

Fig. 3. Graph centered on (a) transactions (b) outputs/inputs (c) linkability

KG structure. The goal of our Petya analysis is to investigate the coin flow between the ransomware address and possible exchanges. As briefly mentioned in the previous section, depending on the task, different views of the blockchain as graphs are preferable. Figure 3 visualizes three different views. In (a) we show the transaction graph of the blockchain, which we use to calculate the fees and can be directly extracted from the blockchain data, in (b) we show the output-input graph with all possible edges of a single transaction, which can be inferred from the blockchain data and in (c) we show the *link graph*, where outputs and inputs are linked if there is a known flow of coins between inputs and outputs of a transaction, which has to be estimated from the blockchain data. These three views are complementary, since in case of coin flow analysis, we are required to know general properties of a transaction, as shown in (a)—e.g., fees, input and output volumes for weighing the coin flow—as well as the actual flow of the coins. In order to capture the flow, we introduce possible "edges" (b) between the inputs and outputs of the transaction, since this information is not present in the blockchain. Note that the link graph (c) is an *uncertain* graph, as the available information in the blockchain is only sufficient to predict such links with a certain likelihood (e.g., via linkability discussed in the preliminaries).

Algorithm 1 Fee Distribution

$$TxOut(t,i,v), ov = sum(v) \rightarrow OutVolume(t,ov) \quad (1)$$

$$TxIn(t,i,st,si), TxOut(st,si,v), iv = sum(v) \rightarrow InVolume(t,iv) \quad (2)$$

$$InVolume(t,iv), OutVolume(t,ov), f = iv - ov \rightarrow Fee(t,f) \quad (3)$$

$$TxIn(t,i,st,si), TxOut(st,si,v), InVolume(t,iv), Fee(t,f), if = (v/iv) * f \rightarrow InFee(t,i,if) \quad (4)$$

Algorithm 1 presents the concept of "fee". Rules 1 and 2 define the total input and output volume per transaction. Rule 3 uses these values to define the fee. Rule 4 assigns to each input of the transaction the weighted fee according to the ratio between the input value and the total input volume.

Algorithm 2 Mapping from Blockchain View to Output-Input (Oi) View

$$TxIn(t, i, st, si) \rightarrow OiNode(st, si, t, i) \quad (5)$$

$$OiNode(ot, oi, it, ii) \rightarrow \exists \text{id } OiId(id), OiOut(id, ot, oi), OiIn(id, it, ii) \quad (6)$$

$$OiIn(id1, t, ii), OiOut(id2, t, oi) \rightarrow PossibleEdge(t, ii, oi), PossibleOiEdge(id1, id2) \quad (7)$$

Algorithm 2 presents the conversion between the blockchain view and the output view. Rule 5 rewrites the blockchain in- and outputs identified by its transaction and index to create a combined view of them. Rule 6 assigns each output-input node a unique id and splits it into an output and input part. Rule 7 creates a possible edge between in and outputs of a single transaction for the output-input view as well as the blockchain view.

Linkability. As described, we extend the possible edges of the output-input-view (b) of the Knowledge Graph with some metric that describes the likelihood of an existing coin flow from an input to an output. The metric we choose here is linkability (as discussed in Section 2). For this, we compute the linkability with Boltzmann for each transaction, where a calculation is feasible in time. We use the default timeout value of 600 seconds and limit the maximum to 16 input or output values (the default is 12). Algorithm 3 shows the application of the output of the Boltzmann computation (i.e., a value between 0 and 1 for each address combination), where we define only edges larger than or equal to the threshold value of the linkability score, here 100%, as a link.

Algorithm 3 Linkability-based Link View

$$PossibleOiEdge(id1, id2), OiAddr(id1, a1), OiAddr(id2, a2), OiIn(id1, t, \_), Linkability(t, a1, a2, l), l \geq 1.0 \rightarrow Link(id1, id2) \quad (8)$$

Note that we allow to include the computation of external tools in the reasoning process (here with accessing the *Linkability* fact), allowing an interaction with highly optimized, task-specific tools. However, the usage of such tools causes a trade-off between explainability and performance, as such computations provide us only with the output value (i.e., a black-box algorithm). For interested readers we provide excerpts of a non-trivial, fully explainable (white-box) approach, expressed in a rule-based format, in the appendix.

Exchange identification. Online sources provide some, but not nearly all addresses of exchanges. Therefore, our goal is to identify additional exchanges based on existing exchange data.

For this, we analyze different transactions where some inputs are from exchanges. Consider for example the transaction "0f44a800..." presented in Figure 4, which is part of our Petya graph. The figure shows that all inputs have a 2-of-3 multisig format (i.e., two of three users have to sign the input that it is accepted in the transaction) and that four of these inputs are tagged by a

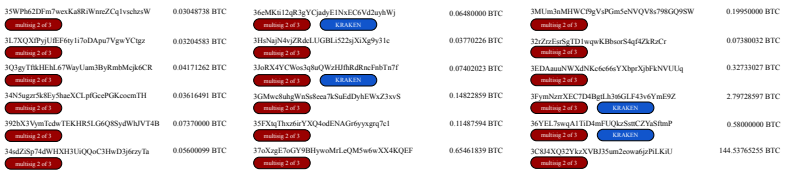

Fig. 4. Transaction, where all inputs have a 2-of-3 multisig format

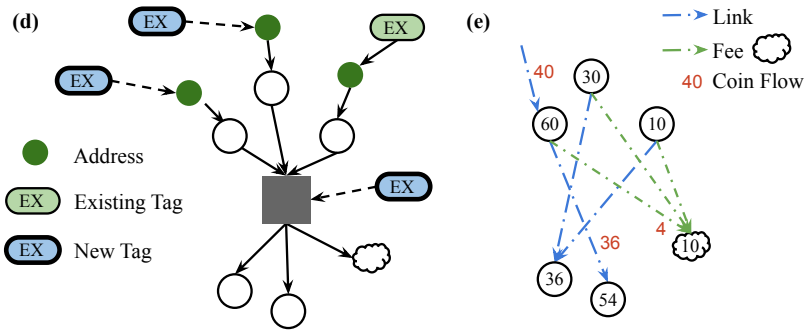

Fig. 5. Graph centered on (d) exchange tags (e) coin flow

specific exchange which we shall refer to as "exchange X", which is an exchange provider for cryptocurrencies. This transaction reveals that there exist some exchange transactions which follow a specific format. In this case, we assume that all inputs are from "exchange X", because they share the same input-type, e.g., 2-of-3 multisig. Therefore, we define a generalized heuristic:

If a transaction has several inputs and one input is tagged as exchange, then all inputs have to be tagged as exchange.

The idea is that it is unlikely that an exchange collaborates with a different party to send transactions. This heuristic allows us to cut the reasoning space by reducing the number of relevant nodes.

The heuristic is visualized in Figure 5 (d), where an existing tag propagates this information to the transaction and to the other addresses of the input nodes.

Algorithm 4 Exchange Tagging

$$TxIn(t,i,st,si), TxOutAddr(st,si,a), Exchange(a) \rightarrow ExchangeTx(t) \quad (9)$$

$$TxIn(t,i,st,si), ExchangeTx(t), TxOutAddr(st,si,a), \rightarrow Exchange(a) \quad (10)$$

Algorithm 4 presents this in rules. Rule 9 models that every transaction having as input an address that belongs to an exchange is an exchange transaction; Rule 10 tags each address of the input of such a transaction as exchange.

Table 1. Statistics of the constructed Petya Knowledge Graph

Nodes in the graph	102170
Edges in the graph	768381
Exchanges in the graph	25253
Bitcoins received at ransomware address	4.3907926
Bitcoins sent at ransomware address	4.03551
Bitcoins not sent at ransomware address	0.355282
Transactions in the graph	1958
Transactions in the graph with linkability score	1514
Transactions in the graph without linkability score	444

4.3 Petya Analysis

In this section, we provide first a general overview of some interesting statistical properties followed by a detail analysis of the coin flow.

Statistical properties. Our goal is to get an overview of the KG, before going into detailed tasks. For this, stakeholders asked the following questions:

- How many coins have been received and spent by the ransomware address? This question indicates the number of relevant coins for analysis.
- For how many transactions are not we able to compute the linkability? Can the linkability computation failure be associated with recurring patterns? These questions indicate the performance of the linkability computation.

Table 1 summarizes the answers to our questions. Our graph has about 100.000 nodes, thereof are about 25% exchanges, with about 770.000 edges between the nodes. Petya collected about 4.4 Bitcoins in the chosen block-interval, whereof about 92% have been moved and thus can be tracked in the coin flow analysis.

To answer the second question, we see in Table 1 that we were not able to compute the linkability of 23% of the transactions. In a detailed analysis, we analyzed these transactions to identify clusters in input and output combinations. We identified that 286 transactions have 2 outputs (i.e., 64.4% of the total transactions without linkability). We use this finding, to improve the linkability in the graph by following "simplified" Bitcoin flow heuristic [6]:

A transaction with two outputs is a payment transaction, where one value is the payment and one the change value and thus both outputs are linked with 100%.

We use this heuristic to extend our "links" in the following coin flow analysis.

Coin flow analysis. The goal of this analysis is to identify the coin flows of the Petya ransomware. For this, we ask following questions:

- How many coins have reached a known point, e.g., exchange, with 100% (75%, 50%) linkability? This is interesting, since exchanges (at least where regulated) are subject to KYC rules. In addition, exchanges are a natural point to stop the coin flow analysis as the ownership of the coin changes.
- How many coins could be additional tracked with the exchange heuristic, how many with the linkability heuristic? This gives us feedback about the improvement contributed by our defined heuristics.

– How many Bitcoins have been spent as fee or got intractable? These values are interesting to complete the overview of the coin flow. Fees are determined by the difference of the input and output of a transaction and get paid to a unrelated account (i.e., the account mining the block) which account is not further relevant for the analysis and are "lost". Intractable coins cannot identify the user due to successfully mixing techniques, but are still of interest.

Algorithm 5 Coin Flow

$$Link(id_1, id_2), OiVal(id_2, val) \rightarrow VLink(id_1, id_2, val) \quad (11)$$

$$VLink(id_1, \_, v), s = sum(v) \rightarrow OutVolume(id_1, s) \quad (12)$$

$$OiDepth(id, 0), OiVal(id, v_{sent}) \rightarrow VNode(id, v_{sent}) \quad (13)$$

$$\begin{aligned} &VNode(id_1, v_{sent}), Link(id_1, id_2), OutVolume(id_1, v_{ov}), \\ &InFee(id_1, v_{fee}), OiVal(id_1, v_{in}), OiVal(id_2, v_{out}), \\ &sentR = v_{sent}/v_{in}, totalSent = v_{sent} - (v_{fee} * sentR), \\ &receiveR = v_{out}/v_{ov}, v_{recv} = totalSent * receiveR \rightarrow VNode(id_2, v_{recv}) \end{aligned} \quad (14)$$

In order to calculate the coin flow, we use a graph traversal query, where we follow all paths with a threshold value (e.g., 100%) of the linkability. Since some paths contain also other inputs, the actual value transferred in an transaction may be higher than the actual value currently flowing through this path. We use weighted fees to approximate the fees of the input and define the following heuristic to approximate the flow value to the output:

The flow to an output is weighted according to all relevant outputs of the input. An output is relevant, if the linkability is higher or equal as the threshold.

Consider Figure 5 (e). We have three inputs `i1` of 60, `i2` of 30 and `i3` of 10 coins, two outputs, `o1` of 36 and `o2` of 54 coins, a linkability of $i1$ with 0.0 for `o1` and 1.0 for `o2` and a input flow of `i1` of 40. Then the fee is 10 coins, whereof input `i1` pays 6 coins, `i2` 3 coins, and `i3` 1 coin. The input flow is $40/60 = 2/3$ and hence the fee of the input flow of `i1` is 4 coins. The total amount of relevant output is 54, and the weighted output of `o1` is $54/54 = 1.0$. Then the flow from `i1` to `o1` is $(40 - 4) * 1.0 = 36$ coins.

Algorithm 5 presents the coin-flow computation. Rule 11 annotates each link with the value of the output, which is used in Rule 12 to calculate the output volume of all valid outputs. Rule 13 initializes the nodes at depth 0 (i.e., transactions to the ransomware address) with the transferred value. Rule 14 propagates this value recursively to the following nodes according to the defined heuristics.

Table 2 provides an overview of the results. We see that 96.7% of the value is tractable (89% to an exchange and 7.7% as fee), which leaves only 3.3% of the coins intractable. While both heuristics improved the result, we see that the exchange heuristic has a much higher impact on the result and increased the coins, which flow to exchanges, from 84.3% up to 88.9%.
Even without the two heuristic we could track more than 84% of the coins to exchanges. This can be explained by an exchange at depth 3, which was used

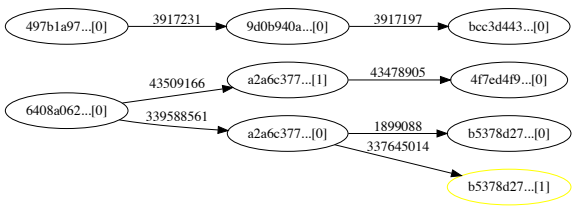

Fig. 6. Fund movements to the high value exchange.

Table 2. Coin Flow Statistics

Bitcoin sent to exchanges	3.5928
Bitcoin spent as fee	0.308909
Bitcoin intractable	0.133801
Bitcoin sent to exchanges (linkability 50%)	3.5928
Bitcoin sent to exchanges (w/o exchange heuristic)	3.40271
Bitcoin sent to exchanges (w/o linkability heuristic)	3.586

to cash out 83.7% of all the coins. Figure 6 visualizes this fund movement by showing the movements after the first consolidations at the lowest depth until the discussed exchange.

We would like to point out that the results are derived using mathematical relationships of Bitcoin transactions and heuristics given by domain experts, so we can guarantee accuracy only with respect to these heuristics. The main goal of this research is to demonstrate how rule-based systems can be used for blockchain analytics, with an emphasis on their ease of use for analytical reasoning tasks and seamless integration with sophisticated algorithms.

5 Related Work

The related work in the area of blockchain analytics is manifold, as multiple problems have been studied in detail such as clustering of blockchain identities [24], de-anonymizing users [20], computation of economic indicators [22], for example to predict future Bitcoin prices [2] or transaction fees, or the identification of criminal activities like ransomware payments [17] or financial fraud [18]. We focus in the following on well-established tools and provide an overview on related ransomware analytics.

Bartoletti et al. [3] created a general Scala library for blockchain analytics focusing on common tasks such as address clustering or user deanonymization to generate a shared view between them. BlockSci [12] is an in-memory database optimized for blockchain analytics for Bitcoin-like blockchains that provides ten pre-defined heuristics for linking addresses and an interface for map-reduce jobs. GraphSense [9, 10] builds upon BlockSci and provides an analytics platform on a NoSQL-basis working on higher-level graph structures such as on the address (i.e., all transactions between two addresses) or entity (i.e., a clustering of ad-

dresses) level and supports the collaborative collection of address tags. Typical clustering heuristics include the multiple-input heuristics which assumes that inputs spent in the same transactions are controlled by the same user (which does not work for mixing transactions where different users are contributing to the input), or the change address heuristic which links the second (lower) output to the input addresses [26]. While those systems focus on a scalable and efficient analytic platform by providing tools, the focus of this work is to study on a concrete scenario the ease of use of rule-based systems for blockchain analytics.

Multiple studies have investigated ransomware activity in the Bitcoin network. One main direction of the work is the detection of addresses related to ransomware [16, 21]. One ransomware studied is CryptoLocker [13, 17], where similar patterns of address usage were detected and a tracking of the coins to a dark web marketplace via an exchange have been reported. Huang et al. [11] estimated a lower bound of ransomware revenue of about $16 millions USD over two years in 2018 and studied the movement of ransomware payments in the Bitcoin network. In comparison, our work focuses on the rule aspect for blockchain analytics and not on new techniques regarding ransomware detection and tracking. Our proposal of adopting Knowledge Graphs can be seen as complementary to existing, highly optimized, task-specific algorithms, bridging the gap between implicitly gained explainability via rules and highly optimized tasks such as clustering algorithms, which can perfectly co-exist (as mentioned in Section 4.2). To the best of our knowledge, rule-based Knowledge Graphs for blockchain analytics have not been considered so far.

6 Conclusion

In this paper, we showed how the key industrial challenge of tracking illicit activities in blockchains can be solved via a rule-based, declarative AI approach. We discussed our methodology for the entire life cycle of the Knowledge Graph, including different views on the blockchain data, integration of third-party tools and a analytical scenario.

Importance and impact. It is a key mission for the financial system to prevent being exploited for illicit activities. For this reason, being able to detect and highlight the risks in new systems, such as blockchains, is crucial to its correct operations. The approach showed impact in a variety of aspects:

- Simple integration of different tools to use in analysis
- Ease of trying out new ideas and using explainable, rule-based formulation for interaction between stakeholders.
- Low barrier of entry, as only high-level technical knowledge is required, and more people can use and *understand* the platform.

Our approach allows fully recursive blockchain queries in a declarative way, which increases the maintainability of blockchain analytics scripts. In future work, we want to improve our system by considering specific performance optimizations for acyclic blockchain graphs, extend the Knowledge Graph to the complete Bitcoin blockchain and develop or use more sophisticated heuristics to improve the overall result.

Acknowledgements

The financial support by the Vienna Science and Technology Fund (WWTF) grant VRG18-013 is gratefully acknowledged.

References

1. Abrams, L.: Petya ransomware's encryption defeated and password generator released. https://www.bleepingcomputer.com/news/security/petya-ransomwares-encryption-defeated-and-password-generator-released/ (2016), [Online; accessed 2021-07-29]
2. Akcora, C.G., Dey, A.K., Gel, Y.R., Kantarcioglu, M.: Forecasting bitcoin price with graph chainlets. In: PAKDD (2018)
3. Bartoletti, M., Lande, S., Pompianu, L., Bracciali, A.: A general framework for blockchain analytics. In: SERIAL@Middleware (2017)
4. Bellomarini, L., Fakhoury, D., Gottlob, G., Sallinger, E.: Knowledge graphs and enterprise AI: the promise of an enabling technology. In: ICDE (2019)
5. Bergman, M.K.: A common sense view of knowledge graphs. https://www.mkbergman.com/2244/a-common-sense-view-of-knowledge-graphs/ (2019), [Online; accessed 2020-04-19]
6. bitcoin.it: Change. https://en.bitcoin.it/wiki/Change, [Online; accessed 2020-04-19]
7. Christen, P.: Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Springer (2012)
8. CoinMarketCap: Total cryptocurrency market cap. https://coinmarketcap.com/charts/ (2021), [Online; accessed 2021-07-29]
9. Haslhofer, B., Karl, R., Filtz, E.: O bitcoin where art thou? insight into large-scale transaction graphs. In: SEMANTiCS (2016)
10. Haslhofer, B., Stütz, R., Romiti, M., King, R.: Graphsense: A general-purpose cryptoasset analytics platform. CoRR **abs/2102.13613** (2021)
11. Huang, D.Y., Aliapoulios, M.M., Li, V.G., Invernizzi, L., Bursztein, E., McRoberts, K., Levin, J., Levchenko, K., Snoeren, A.C., McCoy, D.: Tracking ransomware end-to-end. In: IEEE Symposium on Security and Privacy. pp. 618–631. IEEE Computer Society (2018)
12. Kalodner, H.A., Möser, M., Lee, K., Goldfeder, S., Plattner, M., Chator, A., Narayanan, A.: Blocksci: Design and applications of a blockchain analysis platform. In: USENIX Security Symposium. pp. 2721–2738. USENIX Association (2020)
13. Kharraz, A., Robertson, W.K., Balzarotti, D., Bilge, L., Kirda, E.: Cutting the gordian knot: A look under the hood of ransomware attacks. In: DIMVA. Lecture Notes in Computer Science, vol. 9148, pp. 3–24. Springer (2015)
14. LaurentMT: A standard interface for tokens. https://github.com/Samourai-Wallet/boltzmann, [Online; accessed 2020-04-19]
15. LaurentMT: Bitcoin transactions & privacy (part 1). https://gist.github.com/LaurentMT/e758767ca4038ac40aaf (2017), [Online; accessed 2020-04-19]
16. Li, Y., Cai, Y., Tian, H., Xue, G., Zheng, Z.: Identifying illicit addresses in bitcoin network. In: BlockSys. Communications in Computer and Information Science, vol. 1267, pp. 99–111. Springer (2020)
17. Liao, K., Zhao, Z., Doupé, A., Ahn, G.: Behind closed doors: measurement and analysis of cryptolocker ransoms in bitcoin. In: eCrime (2016)

18. Möser, M., Böhme, R., Breuker, D.: Towards risk scoring of bitcoin transactions. In: Financial Cryptography Workshops (2014)
19. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf (2008), [Online; accessed 2019-11-19]
20. Ober, M., Katzenbeisser, S., Hamacher, K.: Structure and anonymity of the bitcoin transaction graph. Future Internet (2013)
21. Paquet-Clouston, M., Haslhofer, B., Dupont, B.: Ransomware payments in the bitcoin ecosystem. J. Cybersecur. **5**(1), tyz003 (2019)
22. Ron, D., Shamir, A.: Quantitative analysis of the full bitcoin transaction graph. In: Financial Cryptography (2013)
23. Solon, O., Hern, A.: Petya'ransomware attack: what is it and how can it be stopped. The Guardian (2017)
24. Spagnuolo, M., Maggi, F., Zanero, S.: Bitiodine: Extracting intelligence from the bitcoin network. In: Financial Cryptography (2014)
25. Taskar, B., Wong, M.F., Abbeel, P., Koller, D.: Link prediction in relational data. In: NIPS (2003)
26. Zhang, Y., Wang, J., Luo, J.: Heuristic-based address clustering in bitcoin. IEEE Access **8**, 210582–210591 (2020)

A Rule-based Linkability Computation

Assume, we already have created a list of valid combinations by starting with the parent of all inputs linked to all outputs, followed recursively by all non-intersecting pairs of valid input-output sets (i.e., a set is valid if the difference between input volume and output volume is greater or equal than zero and smaller or equal than the fee), which union of input sets and output sets match the parent. Such a list is presented in Table 3 for four inputs with the values (30, 6, 3, 3) and two outputs with the values (10, 15) with the according max depth of each child.

Table 3. Example of Linkability Calculation

Left Child		Right Child		Depth	Map.	Use	Left Child		Right Child		Depth	Map.	Use
{}	{}	{i0,i1,i2,i3}	{o0,o1}	0	16	1	{i1,i2,i3}	{o0}	{i0}	{o1}	1	1	1
{i3}	{}	{i0,i1,i2}	{o0,o1}	1	5	1	{i2}	{}	{i0,i1}	{o0,o1}	2	2	1
{i0,i3}	{o0,o1}	{i1,i2}	{}	1	2	1	{i0,i2}	{o1,o2}	{i1}	{}	2	1	1
{i1,i3}	{}	{i0,i2}	{o0,o1}	1	2	1	{i1,i2}	{}	{i0}	{o0,o1}	2	1	1
{i2,i3}	{}	{i0,i1}	{o0,o1}	1	2	1	{i2}	{}	{i1}	{}	2	1	1
{i0,i1,i3}	{o0,01}	{i2}	{}	1	1	1	{i2}	{}	{i0}	{o0,o1}	2	1	1
{i0,i2,i3}	{o0,o1}	{i1}	{}	1	1	1	{i1}	{}	{i0}	{o0,o1}	3	1	2
{i1,i2,i3}	{}	{i0}	{o0,o1}	1	1	1							

For calculating the linkability, we have to calculate the number of valid mappings and the actual number of usages of each (partial) mapping. This calculation is shown in Figure 7, where (a) counts the number of mappings and (b) counts the number of usages. These numbers are used to calculate the final linkability between each input and output value by counting each left child, where the input

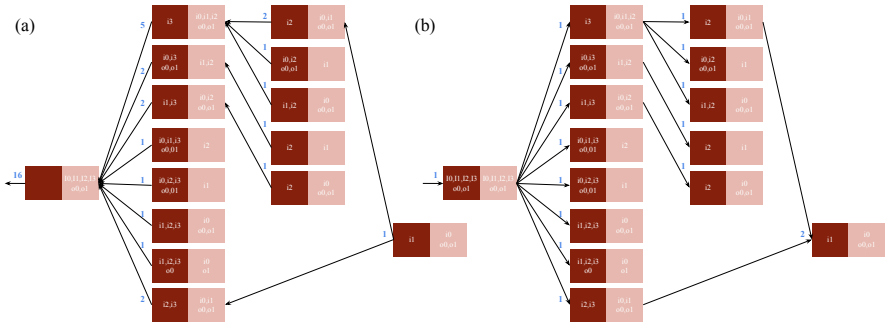

Fig. 7. Calculation of (a) Mappings and (b) Usages

and output value is present, $used * mappings$ times and each right child, where the values are present, $used$ times. The multiplication of the left child considers the number of sub-mappings provided by the right child.

Algorithm 6 defines the number of mappings (a). Rule 1 defines for each element, whether it has a child. Rule 2 uses this information to set the number of mappings to one for all elements that have no children. Rule 3 recursively defines the number of mappings by the sum of the child mappings plus its own mapping.

Algorithm 6 Linkability: Calculation of Mappings

$$List(i_l, o_l, i_r, o_r, \_), List(ci_l, co_l, ci_r, co_r, \_), \\ i_r = ci_l | ci_r, o_r = co_l | co_r \rightarrow HasChild(i_l, o_l, i_r, o_r) \quad (1)$$

$$List(i_l, o_l, i_r, o_r, \_), notHasChild(i_l, o_l, i_r, o_r) \rightarrow Mapping(i_l, o_l, i_r, o_r, 1) \quad (2)$$

$$List(i_l, o_l, i_r, o_r, \_), Mapping(ci_l, co_l, ci_r, co_r, v), \\ i_r = ci_l | ci_r, o_r = co_l | co_r, \\ c = msum(v, < ci_l, co_l, ci_r, co_r >) + 1 \rightarrow Mapping(i_l, o_l, i_r, o_r, c) \quad (3)$$

Algorithm 7 presents the calculation of the number of usages for each mapping (b). For this, we init in Rule 4 the root mapping with 1 and define in Rule 5 recursively the number of usages as the sum of its parents.

Algorithm 7 Linkability: Calculation of Usage per (partial) Mapping

$$List(i_l, o_l, i_r, o_r, 0) \rightarrow Usage(i_l, o_l, i_r, o_r, 1) \quad (4)$$

$$List(i_l, o_l, i_r, o_r, \_), Usage(i_{lp}, o_{lp}, i_p, o_p, v), i_p = i_l | i_r, \\ o_p = o_l | o_r s = msum(v, < i_{lp}, o_{lp}, i_p, o_p >) \rightarrow Usage(i_l, o_l, i_r, o_r, s) \quad (5)$$