BitcoinHeist: Topological Data Analysis for Ransomware Detection on the Bitcoin Blockchain

Cuneyt Gurcan Akcora
University of Texas at Dallas
cuneyt.akcora@utdallas.edu

Yitao Li
University of Texas at Dallas
yitao.li@utdallas.edu

Yulia R. Gel
University of Texas at Dallas
ygl@utdallas.edu

Murat Kantarcioglu
University of Texas at Dallas
muratk@utdallas.edu

Abstract—**Proliferation of cryptocurrencies (e.g., Bitcoin) that allow pseudo-anonymous transactions, has made it easier for ransomware developers to demand ransom by encrypting sensitive user data. The recently revealed strikes of ransomware attacks have already resulted in significant economic losses and societal harm across different sectors, ranging from local governments to health care.**

Most modern ransomware use Bitcoin for payments. However, although Bitcoin transactions are permanently recorded and publicly available, current approaches for detecting ransomware depend only on a couple of heuristics and/or tedious information gathering steps (e.g., running ransomware to collect ransomware related Bitcoin addresses). To our knowledge, none of the previous approaches have employed advanced data analytics techniques to automatically detect ransomware related transactions and malicious Bitcoin addresses.

By capitalizing on the recent advances in topological data analysis, we propose an efficient and tractable data analytics framework to automatically detect new malicious addresses in a ransomware family, given only a limited records of previous transactions. Furthermore, our proposed techniques exhibit high utility to detect the emergence of new ransomware families, that is, ransomware with no previous records of transactions. Using the existing known ransomware data sets, we show that our proposed methodology provides significant improvements in precision and recall for ransomware transaction detection, compared to existing heuristic based approaches, and can be utilized to automate ransomware detection.

I. INTRODUCTION

This decade has been marked with the rise of blockchain based technologies. In its core, blockchain is a distributed public ledger that stores transactions between two parties without requiring a trusted central authority. On a blockchain, two unacquainted parties can create an immutable transaction that is permanently recorded on the ledger to be seen by the public. One of the first applications of Blockchain has been the Bitcoin cryptocurrency [27]. Bitcoin's success has ushered an age known as the Blockchain 1.0 [37], and currently there exist more than 1000 Blockchain based cryptocurrencies.

Bitcoin transactions can be created anonymously, and participation in the network does not require identity verification. A payment can be requested by delivering a public Bitcoin address (i.e., a short string) to a sender by using anonymity networks such as Tor [10]. This ease of usage and worldwide transaction availability of Bitcoin have been noticed by malicious actors as well. The pseudo-anonymity of cryptocurrencies has attracted the interest of a diverse body of criminals, transnational terrorist groups, and illicit users.

Cryptocurrency related crime and, more generally, criminal abuse of blockchain technologies are nowadays recognized as the fastest-growing type of cybercrime [18], [31].

Among the malicious usage of cryptocurrencies, ransomware payments continue to attract an ever increasing attention. Although encrypting files and resources for ransom has a long history, receiving ransom payments securely had never been simple until the emergence of Bitcoin. For example, in 1989 the first documented ransomware AIDS Trojan demanded payments via international money order or cashiers check sent to a P.O. box in Panama. Worldwide transactability of Bitcoin protects ransomware operators from revealing their identity to collect the payments. As Paquet-Clouston et al. [30] state, "The combination of strong and well-implemented cryptographic techniques to take files hostage, the Tor protocol to communicate anonymously, and the use of a cryptocurrency to receive unmediated payments provide altogether a high level of impunity for ransomware attackers". Starting with CryptoLocker in 2013, the world has seen an explosion of ransomware that uses Bitcoin. Many organizations that have been hit by ransomware are asked to pay significant amounts using Bitcoin. For example, Jackson County, Georgia hit with Ryuk in March, 2019 needed to pay bitcoins equal to $400,000, since attackers also compromised the backup data [7].

Furthermore, using cryptocurrenies for ransomware payments appears to be substantially more prevalent than have been previously realized. As noted by Hernendex-Castro et al. [15], among the respondents to their survey, "the prevalence of the Cryptolocker ransomware (3.4%) seems much higher than expected. The proportion of Cryptolocker victims that claim to have agreed to pay the ransom to recover their files (41%) seems to be much larger than expected (3% was conjectured by Symantec, 0.4% by Dell SecureWorks)". Hence, understanding the ransomware payments and its overall economical impact is an emerging challenge of critical societal importance.

Although there have been efforts to analyze the cryptocurrency transactions such as Bitcoin using various heuristics (e.g., "co-spending" heuristic that is based on the idea that input addresses to the same transaction must belong to the same person since private keys associated with those accounts are needed for creating valid transactions [26]) to detect ransomware payments, to our knowledge, none of the previous efforts has leveraged advanced data science tools to detect malicious ransomware payments.

In this paper, *our goal is to identify Bitcoin addresses*

that are used to store and trade Bitcoins gained through ransomware activities. To achieve this goal, we propose a scalable data-driven Bitcoin transaction analytics framework which is substantially more effective in detecting ransomware payment related addresses, compared to the existing heuristic based approaches.

The key thrust behind our proposed approach is the intrinsic capability to observe the complete blockchain graph and, as a result, *to track and analyze dynamics of the associated blockchain topological and geometrical properties at a multi-resolution level.* A natural question arises: *What does local topological blockchain graph structure tells us on anomalous and malicious patterns?* For example, does a suspiciously repeating occurrence of a certain blockchain payment pattern tend to be associated with ransomware payment behavior? We show that by using local topological information available on the Bitcoin transaction graph, we can achieve significant improvement in detecting malicious Bitcoin addresses associated with ransomware payments.

Significance of our contributions can be summarized as follows:

- To our knowledge, this is the first project that leverages advanced data analytics and machine learning techniques for ransomware related bitcoin address detection.
- We propose a simple, tractable and computationally efficient framework to extract features related to Bitcoin transactions which exhibit high utility in predicting ransomware related activities.
- Using the ground truth data collected by various studies, we develop a novel topological data analysis (TDA) based ransomware detection approach that delivers significantly higher precision and recall compared to existing heuristic based procedures.
- In addition to detecting new addresses associated with a known ransomware family, we show that our new methodology could be used to detect the emergence of new ransomware families.

II. RELATED WORK

The success of Bitcoin [27] has encouraged hundreds of similar digital coins [39]. The underlying Blockchain technology has been adopted in many use cases and applications. With this rapidly increasing activity, there have been numerous studies analyzing the blockchain technology from different perspectives.

The earliest results aimed at tracking the transaction network to locate coins used in illegal activities, such as money laundering and blackmailing [3], [29]. These findings are known as the taint analysis [9].

Bitcoin provides pseudo-anonymity; although all transactions are public by nature, user identification is not required to join the network. Mixing schemes [24], [33] exist to hide the flow of coins in the network. Research articles have shown that some Bitcoin payments can be traced [26]. As a result, obfuscation efforts [28] by malicious users have become increasingly sophisticated.

In ransomware analysis, Montreal [30], Princeton [17] and Padua [8] studies have analyzed networks of cryptocurrency ransomware, and found that hacker behavior can help us identify undisclosed ransomware payments. Datasets of these three studies are publicly available.

Early studies in ransomware detection use decision rules on amounts and times of known ransomware transactions to locate undisclosed ransomware (CryptoLocker) payments [19]. More recent studies are joint efforts between researchers and blockchain analytics companies; Huang et al. identify shared hacker behavior and use heuristics to identify ransomware payments [17]. The authors estimate that 20000 victims have made ransomware payments. However, these studies do not extract features and build machine learning models to detect ransomware payments and families.

Feature extraction has been studied for ransomware detection in specific domains. In software code analysis, Cryptolock inspects ransomware programs and their activity for malicious characteristics [34]. In this line of work, studies on ransomware for mobile devices extract software code features to catch malicious programs [23], [2]. However, these studies are mainly targeted for detecting ransomware before it can infect a system, and do not consider Bitcoin transactions.

III. BACKGROUND AND PRELIMINARIES

A. Ransomware

Ransomware is a type of malware that infects a victim's data and resources, and demands ransom to release them. In two main types, ransomware can lock access to resources or encrypt their content. In addition to computer systems, ransomware can also infect IoT and mobile devices [23].

Ransomware can be delivered via email attachments or web based vulnerabilities. More recently, ransomware have been delivered via mass exploits. For example, CryptoLocker used Gameover ZeuS botnet to spread through spam emails. Once the ransomware is installed, it communicates with a command and control center. Although earlier ransomware used hard-coded IPs and domain names, newer variants may use anonymity networks, such as TOR, to reach a hidden command and control server.

In the case of asymmetric encryption, the encryption key is delivered to the victim's machine. In some variants, the encryption key is created on the victim's machine and delivered to the command center.

Once resources are locked or encrypted, the ransomware displays a message that asks a certain amount of bitcoins to be sent to a bitcoin address. This amount may depend on the number and size of the encrypted resources. After payment, a decryption tool is delivered to the victim. However, in some cases, such as with WannaCry, the ransomware contained a bug that made it impossible to identify who paid a ransomware amount.

B. Bitcoin Graph Model

Largely rooted in the existing network analysis methodology, earlier Bitcoin data analytics techniques approached Bitcoin data by creating a graph that employs only a single type of node. Such analytic procedures are referred to as *transaction and address graph approaches.*

In the *transaction graph approach*, addresses are ignored and edges are created among transaction nodes [12], [32]. Naturally, the transaction graph is acyclic and a transaction node cannot have new edges in the future.

In the *address graph approach* [36], transactions are ignored and edges are created among address nodes. However, Bitcoin does not store input to output coin flows explicitly – all inputs are gathered at the transaction, and directed to output addresses at once. As a result, inputs of a transaction must be connected to all its output addresses, which may create large cliques if too many addresses are involved in a transaction.

Single node type approaches do not provide a faithful representation of the Blockchain data. The loss of information about addresses or transactions seem to have an impact in predictive models [14], [1]. In contrast, we model the Bitcoin graph as a heterogeneous network with two node types: addresses and transactions.

Bitcoin Graph Model We consider a directed weighted graph $\mathcal{G} = (V, E, B)$ created from a set of transactions TX and input and output addresses contained in TX. On $\mathcal{G}$, V is a set of nodes, and $E \subseteq V \times V$ is a set of edges. $B = \{\textbf{Address}, \textbf{Transaction}\}$ represents the set of node types. For any node $u \in V$, it has a node type $\phi(u) \in B$. For each edge $e_{u,v} \in E$ between adjacent nodes u and v, we have $\phi(u) \neq \phi(v)$, and either $\phi(u) = \{\textbf{Transaction}\}$ or $\phi(v) = \{\textbf{Transaction}\}$. That is, an edge $e \in E$ represents a coin transfer between an address node and a transaction node. This heterogeneous graph model subsumes the homogeneous case (i.e., $|B| = 1$), where only transaction or address nodes are used, and edges link nodes of the same type. Here, we focus on the case where each address node is linked (i.e., input or output address of a transaction) via a transaction node to another address node. We use Γ^i_a and Γ^o_a to refer to predecessors (in-neighbors) and successors (out-neighbors) of an address a, respectively.

IV. METHODOLOGY

In this paper, we state the following five questions to analyze ransomware behavior on the Bitcoin blockchain: ❶- What features extracted from the Bitcoin network can be used to detect ransomware behavior? ❷- Does a given ransomware family (e.g., Cryptolocker) show the same behavior on the Bitcoin blockchain over time? ❸- How similar is the behavior of different ransomware operators on the Bitcoin blockchain? ❹- Can we detect Bitcoin ransom payments that are not reported to law agencies or Blockchain Analytics companies? ❺- Based on the information about existing ransomware families at a given time, can we detect the emergence of a new ransomware on the Bitcoin blockchain?

To address questions ❶-❺, we formulate two primary research problems: i) detecting undisclosed payments to addresses that belong to a known ransomware family and ii) predicting the emergence of a ransomware family unknown to the date. We start by stating the notations used in our problem definitions. Symbols used in this manuscript are given in Tab. X.

Let $\{a_u\}_{u \in Z^+}$ be a set of addresses, and let each address a_u be associated with a pair $(\vec{x}_u, y_u)$, where $\vec{x}_u \in \mathcal{R}^D$ is a vector of its features and y_u is its label. That is, depending on a setting, y_u can designate a *white* (i.e., non-ransomware) address or a ransomware address. Furthermore, we associate timestamp t_u to represent the time when the address a_u first appeared in a blockchain transaction. An address can appear in the blockchain multiple times. Let $f_1, \ldots, f_n$ be labels of known ransomware families (see Table XI) which have been observed until time point t, and let f_0 be a label of addresses which are **not** known to belong to any ransomware family and are assumed to be white addresses. Before time point t, if we observe l addresses $a_1, \ldots, a_l$, then we form their $D \times l$-matrix of features $X_t = \{\vec{x}_1, \ldots, \vec{x}_l\}$ and a vector of labels $Y_t = \{y_1, \ldots, y_l\} \in \{f_0, f_1, \ldots, f_n\}$.

We formally define our research problems as follows:

Problem 1 [Existing Family Detection]: Let rs be a known ransomware family of interest. Let $\tilde{Y}_t \subseteq Y_t$ be such that $\forall y_j \in \tilde{Y}_t, y_j \in \{f_0, f_{rs}\}$ and $\tilde{X}_t \subseteq X_t$ be the corresponding matrix of features. (If at time point t, $\tilde{Y}_t \cap \{f_{rs}\} = \emptyset$, increase t such that $\tilde{Y}_t$ contains at least one f_{rs}.) Now, let $\{a_{l+1}, \ldots, a_{l+z}\}$ be a set of addresses whose set of labels $Y_{t'} = \{y_{l+1}, \ldots, y_{l+z}\}$ is unknown, and let $X_{t'} = \{\vec{x}_{l+1}, \ldots, \vec{x}_{l+z}\}$ be a set of their corresponding observed features. Furthermore, let $t' > t$, and $t < \min\{t_{a_{l+1}}, \ldots, t_{a_{l+z}}\}$. The problem is to predict all addresses $a_m \in \{a_{l+1}, \ldots, a_{l+z}\}$ such that $y_m = f_{rs}$, using their currently available set of features $X_{t'}$ and previous history (X_t, Y_t).

Problem 2 [New Family Prediction]: Let rs' be a new, yet unobserved ransomware family, and $f_{rs'}$ be its label. Let (X_t, Y_t) be a pair of the sets of features and labels, respectively, such that at time point t, $\forall y_j \in Y_t, y_j \neq f_{rs'}$.

Now, let $\{a_{l+1}, \ldots, a_{l+z}\}$ be a set of addresses whose set of labels $Y_{t'} = \{y_{l+1}, \ldots, y_{l+z}\}$ is unknown, and let $X_{t'} = \{\vec{x}_{l+1}, \ldots, \vec{x}_{l+z}\}$ be a set of their corresponding observed features. Furthermore, let $t' > t$, and $t < \min\{t_{a_{l+1}}, \ldots, t_{a_{l+z}}\}$. The problem is to predict all addresses $a_m \in \{a_{l+1}, \ldots, a_{l+z}\}$ such that $y_m \notin \{f_0, f_1, \ldots, f_n\}$ and a_m is associated with the new ransomware rs (i.e., $y_m = f_{rs'}$), using their currently available set of features $X_{t'}$ and previous history (X_t, Y_t).

A. Graph features for classification

On the heterogeneous Bitcoin network, the in-neighbors Γ^i_n of a transaction tx_n is defined as the set of transactions (not addresses) whose one or more outputs are input to transaction tx_n. The out-neighbors of tx_n are denoted as Γ^o_n.

A transaction has inputs and outputs; the sum of output amounts of a transaction tx_n is defined as $\mathcal{A}^o(n) = \sum_{a_u \in \Gamma^o_n} A^o_u(n)$, where an output address a_u receives $A^o_u(n)$ coins.

On the Bitcoin network, an address may appear multiple times with different inputs and outputs. An address u that appears in a transaction at time t can be denoted as a^t_u. As most addresses appear only once, for sake of simplicity, we omit t throughout our notations when it is clear from the context. In order to mine address behavior in time, we divide the Bitcoin network into 24 hour long windows by using the UTC-6 timezone as reference. This window approach serves
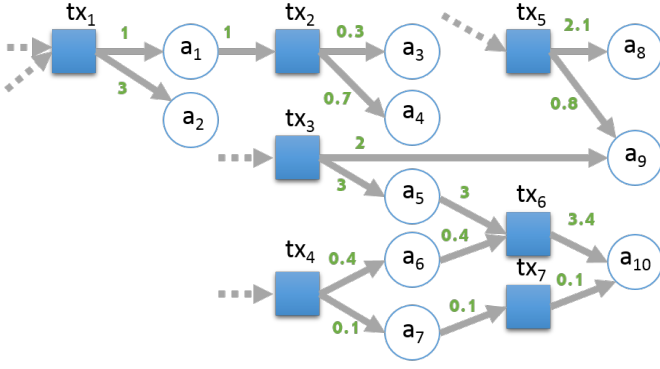
Fig. 1: A toy network of 10 addresses and 7 transactions. Dashed edges indicate transaction outputs from earlier windows; t_1, t_3, t_4 and t_5 are starter transactions. Coin amounts are shown on edges. Transaction outputs are equal to transaction inputs, i.e., transaction fees are 0.

two purposes. First, the induced 24 hour network allows us to capture how fast a given coin moves in the network. The speed is measured by the number of blocks in a the 24 hour window that contain a transaction involving the coin. In maximum, a coin can appear in 144 blocks (24 hours, 6 blocks per hour). Coin speed may disclose certain information on transaction intent. For example, coins that are moved too frequently in the network may be involved in money laundering. Second, the temporal order of transactions within the window helps us distinguish transaction activity from different geolocations. Temporal information of transactions, such as the local time, has been found useful to cluster criminal transactions (see Figure 7 in [17]).

On the heterogeneous Bitcoin network, in each snapshot we extract the following six features for an address: *income, neighbors, weight, length, count, loop*.

Income of an address u is the total amount of coins output to u: $I_u = \sum\limits_{t_n \in \Gamma_u^o} A_u^o(n)$.

Neighbors of an address u is the number of transactions which have u as one of its output addresses: $\left|\Gamma_u^i\right|$.

We define the next four address features by using their time ordered position in the defined 24 hour time window. We denote time of a window with the earliest time t of transactions contained in it. For each window, we first locate the set of transactions that do not receive outputs from any earlier transaction within the studied window t, i.e., $\mathbb{TX} = \{\forall tx_n \in TX, s.t., \Gamma_n^i = \{a_1^{t^o}, \ldots, a_z^{t^n}\}, t^0 \le t^n < t\}$. These transactions consume outputs of transactions that have been generated in previous windows. For simplicity, we refer to a transaction $tx \in \mathbb{TX}$ as a **starter** transaction.

Weight of an address u, W_u, is defined as the sum of fraction of coins that originate from a starter transaction and reach u.

Length of an address u, L_u, is the number of non-starter transactions on its longest chain, where a chain is defined as an acyclic directed path originating from any starter transaction

and ending at address u. A length of zero implies that the address is an output address of a starter transaction.

Count of an address u, C_u is the number of starter transactions which are connected to u through a chain, where a chain is defined as an acyclic directed path originating from any starter transaction and ending at address u.

Loop of an address u, O_u is the number of starter transactions which are connected to u with more than one directed path.

Rationale: Our graph features are designed to quantify specific transaction patterns.

Loop is intended to count how many transaction i) split their coins; ii) move these coins in the network by using different paths and finally, and iii) merge them in a single address. Coins at this final address can then be sold and converted to fiat currency (see Figure 7 in [25] for examples of such patterns).

Weight quantifies the merge behavior (i.e., the transaction has more input addresses than output addresses), where coins in multiple addresses are each passed through a succession of merging transactions and accumulated in a final address (see aggregations in Figure 1 of [17] for a potential application of this pattern).

Similar to weight, the count feature is designed to quantify the merging pattern. However, the count feature represents information on the number of transactions, whereas the weight feature represents information on the amount (what percent of these transactions' output?) of transactions.

Length is designed to quantify mixing rounds [24] on Bitcoin, where transactions receive and distribute similar amounts of coins in multiple rounds with newly created addresses to hide the coin origin (see the mixing rounds in Figure 2 of [33]).

Example 1 (Graph Features): Consider the toy network in Figure 1. We begin from defining the starter transactions $\mathbb{TX} = \{tx_1, tx_3, tx_4, tx_5\}$ as in this specific window, they receive no coins from an earlier transaction.

We assign each of the four transactions a weight of 1, regardless of their output amounts $\mathcal{A}^o(.)$. Addresses a_1 and a_2 receive $W_{a_1} = 1/|\Gamma_{tx_1}^o| = 1/2$ and $W_{a_2} = 1/|\Gamma_{tx_1}^o| = 1/2$ from transaction tx_1, respectively. Address a_3 receives $1/2$ of the weight from a_1. Hence, $W_{a_3} = (1/2) \times (1/2)$. Consider a_{10} which receives weights from tx_3 and tx_4. Weight of tx_3 flows through a_5 and t_6, whereas the weight of tx_4 flows through a_7 and tx_7 to reach a_{10}. Hence, $W_{a_{10}} = W_{a_7} + [W_{a_5} + W_{a_6}] = (1/2) + [1/2 + 1/2]$.

Length of a_1 and a_2 is 0, but $L_{a_3} = L_{a_4} = 1$, since a_4 and a_5 can be reached from tx_1 through a non starter transaction tx_2. Furthermore, a_{10} can be reached from tx_3 or tx_4, hence $L_{a_{10}} = 1$. Although a_8 and a_9 appear later than a_3 and a_4, lengths of a_8 and a_9 are shorter, because these addresses are direct outputs of the starter transaction tx_5.

The address a_{10} has three chains: i) $tx_3 \to a_5 \to tx_6 \to a_{10}$, ii) $tx_4 \to a_6 \to tx_6 \to a_{10}$ and $tx_4 \to a_7 \to tx_7 \to a_{10}$, but these chains start from three starters only (i.e., tx_3 and tx_4). Hence, its count $C_{a_{10}} = 2$.

Transaction tx_4 has two chains that reach a_{10} through tx_6 and tx_7, separately. These two chains form a loop from tx_4 to a_{10}. Hence, $O_{a_{10}} = |\{tx_4\}| = 1$. All other nodes have zero loops.

Feature Standardization. In data pre-processing, we use feature standardization to make the values of each feature have zero-mean and unit-variance. We first compute mean and standard deviation for each feature and subtract the mean from each feature value. Then we divide each feature value by its standard deviation.

B. Ransomware address classification and clustering

Before detailing our TDA model, in this section we first outline four classification and clustering models that we employ in this paper.

Similarity Search:

We use addresses contained in a specific time window t, and compute pairwise similarity to known ransomware addresses from the last l days.

Heuristics

Following two heuristics that are defined by Meiklejohn et al. [26] are used in our experimental evaluation.

Co-spending: "If two addresses are inputs to the same transaction, they are controlled by the same user".

Transition: "If we observe one transaction with addresses A and B as inputs, and another with addresses B and C as inputs, then we conclude that A, B, and C all belonged to the same user".

We did not implement the change heuristic proposed in [26], because addresses discovered by the change heuristic are not guaranteed to belong to the same user.

Unsupervised approaches

DBSCAN density clustering: Density-based spatial clustering of applications with noise (DBSCAN) is a density-based non-parametric clustering algorithm. DBSCAN can mark outlier points that lie alone in low-density regions as noise [11].

Hierarchical clustering: We use k-means clustering with Forgy algorithm based initial seed selection [13].

Tree based approaches

Extreme Gradient Boosting Trees (XGBT): applies gradient boosting algorithms to decision trees [6].

Random forest is a supervised ensemble of multiple simple decision trees to estimate the dependent variables of the data [16].

C. Topological Data Analysis Models: TDA Mapper

In this project we introduce the concepts of topological data analysis (TDA) into detection of ransomware patterns on Blockchain. The fundamental idea of TDA is to extract hidden data patterns via systematic analysis of data shapes such as, cycles and flares, quantified at various resolution scales [5], [21].

TDA offers the following multi-fold benefits which are particularly useful in the context of ransomware detection. First, TDA assesses data shapes in a coordinate-free manner, which implies that we can systematically compare patterns of data obtained under various data collection frameworks. Second, TDA analyzes properties of data shapes that are robust to minor data perturbations. Hence, TDA tends to deliver more consistent results on hidden data patterns even under noisy data collection schemes which is a typical scenario in money laundering studies. Finally, TDA offers a low dimensional description of some key properties of the high-dimensional data by using a finite combinatorial object for systematic extraction of data shape patterns. In this paper, we employ the Mapper method of [35], [5] which is a highly customizable TDA tool allowing for a systematic multi-resolution glimpse into organization and functionality behind the underlying data generating process. By complementing more traditional clustering and projection pursuit approaches with a systematic insight on data geometry and topology, Mapper often recovers hidden data patterns that are otherwise inaccessible with conventional data analytic techniques.

The key idea behind Mapper is the following. Let U be a total number of observed addressed and $\{\vec{x}_u\}_{u=1}^U \in \mathcal{R}^D$ be a data cloud of address features. Select a filter function $\xi : \{\vec{x}_u\}_{u=1}^U \to \mathbb{R}$.

Let I be the range of ξ, that is, $I = [m, M] \in \mathbb{R}$, where $m = \min_u \xi(\vec{x}_u)$ and $M = \max_u \xi(\vec{x}_u)$. Now place data into overlapping bins by dividing the range I into a set S of smaller overlapping intervals of uniform length, and let $u_j = \{u : \xi(\vec{x}_u) \in I_j\}$ be addresses corresponding to features in the interval $I_j \in S$. For each u_j perform a single linkage clustering to form clusters $\{u_{jk}\}$.

To find the number of clusters, Mapper analyzes empirical distribution of edge lengths at which each cluster is merged based on the rationale that internal distances (i.e., within a cluster) are expected to be lower than external distances (i.e., in-between clusters) and distributions of internal and external distances are disjoint. Let $\{u_{jk}\}$ be addressed in the k-th cluster of the j-th interval. View each cluster as a node and draw an edge between two nodes k and m if clusters $\{u_{jk}\}$ and $\{u_{lm}\}$ contain overlapping addresses, i.e., $\{u_{jk}\} \cap \{u_{lm}\} \neq \emptyset$.

As a result, Mapper produces a low dimensional representation of the underlying data structure in the form of "cluster tree" graph $\mathcal{CT}$ where each "cluster" is a branch of some single connected component rather than a disconnected component on its own as in conventional clustering analysis. In Fig. 2, we show an example of the produced Mapper graph. Each node may contain three sets of addresses: past ransomware addresses, past non-ransomware addresses, and addresses of the current time window, whose labels are unknown. If current addresses are contained in clusters that also contain many

Fig. 2: Mapper graph of the Cerber ransomware addresses (2017 day 307), filtered with the length attribute. Around clusters we indicate the number of past ransomware addresses contained in the cluster. We show total order (i.e., address count) of two biggest clusters inside circles (326 and 537). The two clusters with 19 past ransomware addresses (top left) contain 53 and 67 addresses. Clusters without past ransomware are depicted in yellow.

past known ransomware addresses, we deem these current addresses potential ransomware addresses.

We filter the TDA mapper graph of using each of our six graph features. As a result, we obtain six filtered graphs $\mathcal{CT}_1, \ldots, \mathcal{CT}_6$ for each time window. Afterwards, we assign a suspicion, or risk score to an address a_u, as outlined in Algorithm 1.

input : A set of networks $\mathcal{CT}_1, \ldots, \mathcal{CT}_D$; filter threshold q; inclusion threshold ϵ_1; size threshold ϵ_2; set of past ransomware addresses RS; set of past non-ransomware addresses NRS.

output: A set of suspicious addresses.

$P : Map \leftarrow$ Initialize scores of all l addresses with 0;

foreach *network* $\mathcal{CT} \in \{\mathcal{CT}_1, \ldots, \mathcal{CT}_D\}$ **do**
  foreach *cluster* $C_c \in \mathcal{CT}$ **do**
    $A_c \leftarrow$ select all addresses in C_c;
    $V \leftarrow A_c \cap RS$;
    if $|V| \geq \epsilon_1 \times |RS|$ **then**
      if $|A_c| \leq \epsilon_2 \times |\mathcal{CT}.V|$ **then**
        foreach $a_u \in A_c \setminus \{RS \cup NRS\}$ **do**
          $P_u \leftarrow 1 + P_u$;

$q_t \leftarrow quantile(P, q)$;
$S = \{\forall a_u \in P | P_u \geq q_t\}$;
return S;

Algorithm 1: TDA filtering with multiple attributes.

Algorithm 1 starts by computing the number of past

Fig. 3: Address recurrence in days. 11 addresses have more than 120 appearances (omitted here).

ransomware addresses in each cluster. If both inclusion and size thresholds, ϵ_1 and ϵ_2, respectively, are satisfied, addresses in the cluster have their suspicion scores incremented.

Parameters. We use two parameters to control what we learn from mapper clusters: inclusion and size parameters. The inclusion parameter ϵ_1 limits what can be learned when very few ransomware addresses are contained in the cluster. The size threshold ϵ_2 prevents learning when cluster includes too many addresses. Such phenomenon usually happens if a filtering feature does not exhibit a sufficiently discriminating performance during a specific time window, and all addresses are lumped together. For example, in Figure 2 the largest cluster contains 78 past ransomware addresses. Given this finding and the cluster size of 537 addresses, the risk scores of the remaining 459 addresses are to be elevated – that is, the procedure results in testing of too many addresses as potentially suspicious. We further use a quantile threshold q on addresses, and label addresses suspicious only if they are in the top $1-q$ of all addresses. We emphasize that by controlling q, ϵ_1 and ϵ_2 parameters, we can avoid making predictions when evidence of past ransomware is not sufficiently *strong*.

We will denote TDA models with the $\mathbb{TDA}_q^{\epsilon_1|\epsilon_2}$ notation. TDA models may deliver nested results; for example $\mathbb{TDA}_q^{0.5|\epsilon_2}$ may return the same set of suspicious addresses as $\mathbb{TDA}_q^{0.7|\epsilon_2}$ results. For such cases, we prefer the most restrictive model; i.e., the model with the highest q, highest ϵ_1 and lowest ϵ_2.

In summary, the key benefit of the Mapper approach is its capability to recover hidden similarities (i.e., connections) between these "clusters", or groups of addresses, that are typically unavailable with traditional clustering techniques.

V. RANSOMWARE BEHAVIOR

Dataset. We have downloaded and parsed the entire Bitcoin transaction graph from 2009 January to 2018 December. Using a time interval of 24 hours, we extracted daily transactions on the network and formed the Bitcoin graph. We filtered out the network edges that transfer less than ฿0.3, since ransom amounts are rarely below this threshold. As mentioned in Sec. VII, in final verification we used the full graph to discover smaller amounts that we may have missed due to this filtering.

Our ransomware dataset is a union of datasets from three widely adopted studies: Montreal [30], Princeton [17] and Padua [8]. The combined dataset contains 24,486 addresses

from 27 ransomware families. From these addresses, we extract the features listed in Tab. XI. The column N shows the number of times addresses from a given family appear in the Bitcoin blockchain. The *unique* column gives the number of unique addresses for each family. In 24 ransomware families, at least one address appears in more than one 24-hour time window. Fig. 3 shows a histogram of appearances. CryptoLocker has 13 addresses that appear more than 100 times each. The CryptoLocker address 1LXrSb67EaH1LGc6d6kWHq8rgv4ZBQAcpU appears for a maximum of 420 times. Four addresses have conflicting ransomware labels between Montreal and Padua datasets. APT (Montreal) and Jigsaw (Padua) ransomware families have two and one *multisig* addresses, respectively. All other addresses are ordinary addresses that start with '1'.

A. A summary on ransomware features

TABLE I: Most common 6 feature value combinations (i.e. patterns) in ransomware addresses. Address column gives the number of addresses with the pattern. Rank of a pattern is computed over all (ransomware or not) addresses in the blockchain. Difference in ranks of ransomware and non-ransomware patterns is found statistically significant by two-sample Wilcoxon rank sum test [4], p-value $< 2.2 \times 10^{-16}$.

Length	Weigth	Neighbor	Count	Loop	Income	#Address	Rank
0	0.5	2	1	0	₿1	327	1
0	0.5	2	1	0	₿1.2	250	113
0	1.0	2	1	0	₿1	189	4
0	1.0	1	1	0	₿0.5	178	9
0	0.5	2	1	0	₿0.8	160	116
0	1.0	1	1	0	₿1	146	3
0	1.0	2	1	0	₿1.2	127	121
0	0.5	2	1	0	₿1.25	119	327
0	0.5	1	1	0	₿0.5	118	6
0	1.0	1	1	0	₿2	117	18

We compute six feature values of 48,168 ransomware address appearances. We find 30K unique feature values (i.e., unique rows of data), which we term **patterns**. Table I lists the top 10 most frequent patterns in ransomware address features. Here, 54 of the top 100 most frequent patterns of all addresses are also found in the top 100 of ransomware patterns. The *Rank* column shows the rank of a pattern in all transactions. We find that four of the top-10 ransomware address patterns do not appear in the top-100 patterns of all addresses. Furthermore, length=0 implies that ransomware addresses receive payments from starter transactions; in the given window, such addresses have no past history. Finally, weight 0.5 implies that the starter transaction pays the ransomware address, and uses a change address. We also find that a ransomware address typically receives all coins of the transaction, i.e., weight=1.0.

Table XI shows mean values of the graph features for ransomware addresses. CryptoLocker, CryptXXX, Locky, Cerber, DMALockerv3 and CryptoWall have more than 100 unique addresses. The Razy ransomware exhibits an outlier behavior in five features. In weight, only DMALocker has a value > 1. WannaCry family is an outlier in both length and loop features. Figure 4 depicts distributions of two features, count and loop. When compared to non-ransomware addresses, ransomware addresses exhibit more profound right skewness in distributions of feature values.

B. Feature Relevance for ransomware behavior

We compute feature values for each appearance of ransomware addresses, and visualize the six-dimensional feature data in Fig. 5 for the six largest families. Each data point is given a location in a two-dimensional map by the t-stochastic neighborhood embedding [22]. We show results for three perplexity values; perplexity "can be interpreted as a smooth measure of the effective number of neighbors" [22]. Distances and sizes may not be meaningful in TSNE, however, groups of data points exhibit interesting patterns. Each family can be seen to have small groups of co-clustering addresses, implying that a few addresses have similar features. As perplexity increases, we find that addresses tend to cluster together, with DMALocker and CryptXXX addresses appearing at the center. CryptoWall and CryptoLocker addresses appear together in many groups. The most important insight to be gained from TSNE results is that addresses from a ransomware family do not all exhibit the same behavior – each family have a multitude of patterns that repeat across addresses.

C. Ransomware behavior similarity

We use ransomware labels as external ground truth, and cluster ransomware addresses in up to 20,000 clusters. We compute cluster purity for each family, which we define as the percentage of addresses from a family appearing in clusters where every address in the cluster belongs to the same ransomware family. In computing purity, we only consider clusters that contain more than one data point. Fig. 6 depicts purity for the eight biggest ransomware families by address count; Cerber CryptoWall and CryptoLocker reach almost 40% purity. As in Fig. 6, we find an optimal number of $k = 12500$ clusters.

Next, we analyze the resulting clusters for relationships between ransomware families. In Tab. II, we show the number of clusters that are shared by ransomware pairs. CryptoWall addresses appear in 1152 pure clusters (all addresses belong to CryptoWall). CryptoLocker and CryptoWall addresses appear in 2587 impure clusters. In Tab. III, we further look into these shared clusters and count how many addresses are co-clustered in families. We see that 5122 CryptoLocker addresses (1st row, 2nd column) are clustered with 4826 CryptoWall addresses (2nd row, 1st column). From Tab. III, we again see that CryptXXX more often co-clusters with CryptoLocker. The case of CryptoLocker-CryptoWall is intriguing; CryptoLocker addresses cluster more often with CryptoWall addresses than with other CryptoLocker addresses.

In August 2014, the CryptoLocker ransomware was taken down by Operation Tovar: a consortium constituting FBI, Interpol, security software vendors and several universities. Two months earlier, the CryptoWall ransomware had been discovered. CryptoLocker spread through email attachments, whereas in addition to attachments, CryptoWall uses kits hosted on compromised websites or malicious ads. CryptXXX appeared in April 19, 2016.

Clustering uses address features that encode transaction behaviors of two set of users: initial ransom payers, and hackers that move paid ransom amounts on the network. As initial payers are unknown to each other, we do not expect them to exhibit similar behavior. We believe that the similarity implied by the
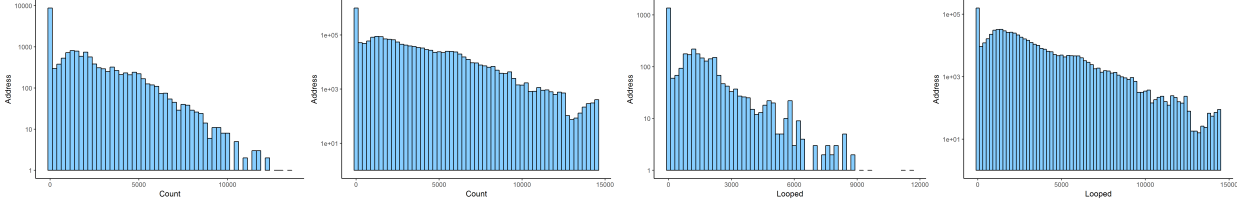
(a) Count distribution in ransomware addresses.

(b) Count distribution in non-ransomware addresses.

(c) Loop distribution in ransomware addresses.

(d) Loop distribution in non-ransomware addresses.

Fig. 4: Distributions of feature values for ransomware and non-ransomware addresses. Ransomware addresses are skewed for both features. The skew also exists in weight and length features (not shown here).
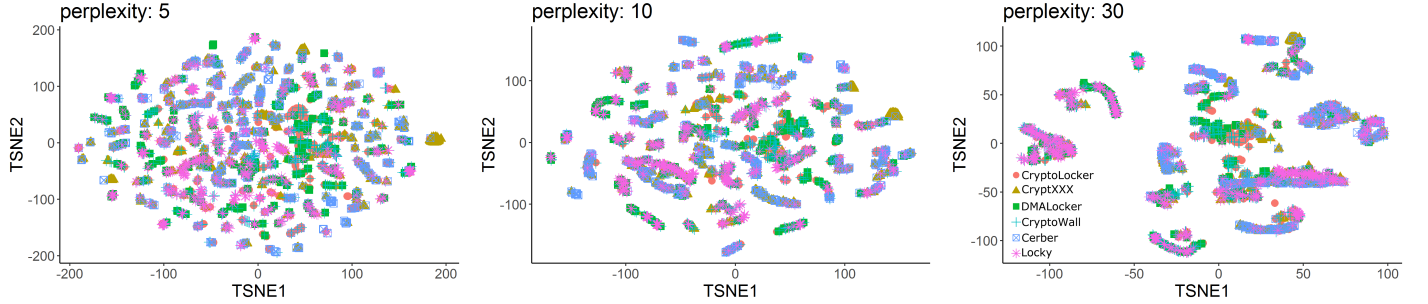

Fig. 5: Two dimensional t-Stochastic Neighbor embeddings for addresses from six ransomware families. The perplexity can be interpreted as a measure of the effective number of neighbors to be considered for each point.
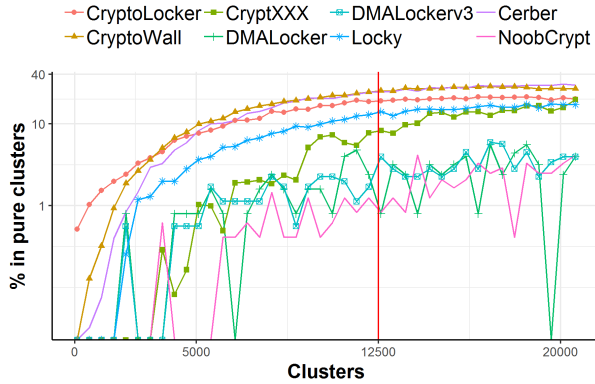
Fig. 6: Clustering ransomware addresses by using six features.

TABLE II: Number of shared clusters between ransomware families ($k = 12500$). At least one address of the ransomware in row is co-clustered with at least one address of the ransomware in columns. If more than two families exist in a cluster, the cluster is counted for each pair.

	Crypto Wall	Crypto Locker	Cerber	Locky	Crypt XXX	Noob Crypt	DMA Locker
CryptoWall	1152	2587	976	1403	374	304	129
CryptoLocker		690	1220	841	379	142	99
Cerber			647	493	562	18	20
Locky				296	266	87	55
CryptXXX					49	24	5
NoobCrypt						6	28
DMALocker							2

TABLE III: Number of co-clustering addresses from ransomware families ($k = 12500$).

	Crypto Wall	Crypto Locker	Cerber	Locky	Crypt XXX	Noob Crypt	DMA Locker
CryptoWall	3145	5122	1737	3015	624	702	321
CryptoLocker	4826	1766	2380	1843	618	238	179
Cerber	3415	2285	1885	9763	1696	22	37
Locky	4077	3023	1512	832	460	405	384
CryptXXX	1024	1077	1467	355	155	52	5
NoobCrypt	344	164	18	95	25	12	37
DMALocker	140	107	21	61	6	33	4

clustering results can only be due to shared hacker behavior. As such, CryptoLocker, CryptoWall and CryptXXX may be created and run by the same hackers. As a supporting evidence, for CryptXXX, the security company Symantec warns that "Definitions *(of the detection tool)* prior to September, 2016 may detect this threat as Trojan.CryptoLocker.AN") [38] (We could not find such a note for other ransomware families).

D. Address features over time

In our detection and prediction problems, we assume that ransomware addresses exhibit similar feature patterns in time, and we can learn to identify these patterns. However, the TSNE results in Sec. V-B show that a global behavior may not exist.

In fact, when we compute ransomware patterns, Tab. IV shows that only 10 families have feature patterns that are repeated in time (see Tab. I patterns). For example, Cerber has 3491 addresses use 564 unique patterns more than once. Some of

these pattern repeating addresses appear in hundreds of time windows: the address 1LXrSb67EaH1LGc6d6kWHq8rgv4ZBQAcpU appears in 420 windows. We cluster it in 66 clusters multiple times (as given in Sec. V-C); in one cluster it appears six times. Overall, 100 addresses from CryptoLocker, CryptoWall, NoobCrypt, DMALockerv3, GlobeImposter, DMALocker and CryptXXX families have repeating patterns in time. These results offer evidence that although we cannot talk of a general behavior, small groups of addresses from at least 10 families can be used to discover other ransom addresses.

TABLE IV: Repeating patterns in ransomware transactions.

Ransomware	Unique pattern	# addresses
Cerber	564	3491
Locky	403	2704
CryptoWall	389	1406
CryptoLocker	248	1201
CryptXXX	141	1018
DMALockerv3	19	56
Globev3	3	7
DMALocker	3	6
GlobeImposter	1	2
NoobCrypt	1	2

VI. RANSOMWARE DETECTION AND PREDICTION

In this section we first discuss two factors that we account for in training our models.

Class imbalance. Compared to 24,486 known ransomware addresses from 2013 to 2018, there exist 200K daily transactions on average on the Bitcoin network, where each transaction contains an average of 5 input and output addresses. As a result, we observe a large number of f_0 (i.e, non-ransomware) addresses, leading to a class imbalance problem in classification. To reduce class imbalance, we train on a limited number of addresses; that is, we employ $N \in \{300, 600, 1000\}$ samples of f_0 and $f_1, \ldots, f_n$ (i.e., ransomware) addresses each. Our choice is due to the fact that on most days we observe much fewer than 100 unique ransomware addresses.

Temporal variation. Ransomware families appear and disappear in certain time periods. For example, the CryptoLocker ransomware was active until 2014 September. Although its clones, such as CryptoWall, appeared afterwards, using CryptoLocker addresses in training for later days may increase false positives. To mitigate these effects, we employ a sliding window approach, and use data from last $l \in \{30, 60, 90, 120, 240\}$ windows to train our models.

Metrics. For each model and ransomware family, we compute precision and recall by using overall sums of TN, FN, FP and TP values across multiple time windows. Precision and recall are computed as $TP/(TP + FP)$ and $TP/(TP + FN)$, respectively. As we observe considerably more non-ransomware addresses, we identify the positive likelihood ratio ($PLR = TP/FP$) as an important metric, since PLR quantifies the effort needed for analyzing the blockchain by using model prediction results. For example, a PLR of 1 implies that for every TP address, the analyst has to manually analyze one FP address, unnecessarily.

Parameters. In all models, we report the optimal parameters that maximize F1 scores in predictions as follows: In DBSCAN, we experimented with $\epsilon = 0.05, \ldots, 1$ values. Random

Forest uses ntree=500 and mtry=$|X_t|/3$. XGBoost uses the gbtree booster and nrounds = 25. For TDA computations, we use the TDAMapper RStats package (https://github.com/paultpearson/TDAmapper) with parameters overlap=40, interval = 80.

A. Existing family detection

We outline our experimental settings for the detection problem as follows.

Given features and known labels of past addresses X_t, Y_t at time t and features of addresses $X_{t'}$ at time $t' > t$, we train for existing ransomware family detection as follows:

1) Select a ransomware family rs whose new addresses will be detected at time t'.
2) For $t < t'$, use a training length l, and create a dataset X_t which holds features and labels of addresses observed between times $t - l$ and t.
3) Create an f_0 sample of size N from $X^0_{[t-l,t]} \subseteq X_{[t-l,t]}$ without replacement where $\forall x_u \in X^0_{[t-l,t]}, y_u = f_0$ and $N = \left| X^0_{[t-l,t]} \right|$.
4) Create a *ransomware* sample of size N from $X^{rs}_{[t-l,t]} \subseteq X_{[t-l,t]}$ without replacement where $\forall x_u \in X^a_{[t-l,t]}, y_u = f_{rs}$ and $N \leq \left| X^{rs}_{[t-l,t]} \right|$.
5) Using the ground truth data at t', find all ransomware addresses for t': $X^{rs}_{t'}$.
6) Using the ground truth data at t', take a sample of $M = 1000$ white (i.e., f_0) addresses without replacement: $X^0_{t'}$.
7) Remove past known addresses from $X^{rs}_{t'}$, i.e., $X^{rs}_{t'} \leftarrow X^{rs}_{t'} \setminus X^{rs}_{[t-l,t]}$.
8) Use features $\{X^0_{[t-l,t]} \cup X^{rs}_{[t-l,t]}\}$ and labels $\{Y^0_{[t-l,t]} \cup Y^{rs}_{[t-l,t]}\}$ as the training data, and classify $\{X^0_{t'} \cup X^{rs}_{t'}\}$.

We emphasize four aspects of existing family detection: i) [Step 7]: from the test dataset we remove appearances of addresses that have appeared in the past (i.e., $t < t'$), because their labels are already known. ii) [Step 4]: If an address appears in multiple windows, its each appearance will have (potentially) different features in $X_{[t-l,t]}$ with the same rs label. iii) [Step 7]: On many days, we will not have N past rs addresses to train from. iv) Most importantly, *we learn a model for each ransomware family*. In our experiments, we show that these models do not share the same characteristics.

Heuristics. By employing the co-spending and transition heuristics with all past history (i.e., $N = |X_t|$, and $l = \infty$), we discover only 40 unique addresses from CryptoLocker (Padua), CryptoWall (Padua), CryptoTorLocker2015 (Montreal), CryptoTorLocker2015 (Padua) families.

Naive similarity search. An interesting benchmark for detecting undisclosed payments from existing families is to compute the similarity of $X_{t'}$ addresses to the past addresses in X_t. If existing families exhibit repeating patterns over time, the similarity search can match new addresses to known ransom addresses. In fact, Fig. 7 shows that this strategy may be effective. By using exact matches to known ransom patterns of the past, in six days similarity search reveals more than 50 addresses each day, while predicting a maximum of 73 false
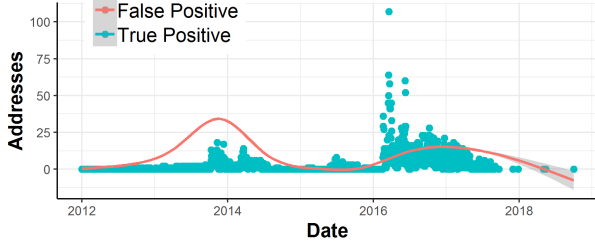
Fig. 7: Ransomware detection with exact feature matches.

positives on 2013 day 336. These results offer supporting evidence of utility of our features. However, this naive approach creates 21,371 FP addresses overall, which makes it unfeasible for operational use by security analysts.

TABLE V: Existing family address detection results.

RS	Method	l	N	TP	FP	FN	TN	#w	Prec	Rec	F1	PLR
Locky	$\text{TDA}_9^{.8|.5}$	240	300	451	2350	50	8221	11	0.161	0.900	0.273	0.192
	COSINE	90	300	2395	41681	3990	146369	194	0.054	0.375	0.095	0.057
Crypto Wall	$\text{TDA}_9^{.8|.65}$	240	600	217	3087	155	11200	15	0.066	0.583	0.118	0.070
	$\text{DBSCAN}_{.2}$	240	600	728	18960	794	16913	59	0.037	0.478	0.069	0.038
Crypto Locker	$\text{TDA}_9^{.65|.65}$	240	300	439	9686	212	22129	34	0.043	0.674	0.081	0.045
	$\text{DBSCAN}_{.15}$	60	300	935	42771	295	11316	67	0.021	0.760	0.042	0.022
Cerber	$\text{TDA}_9^{.5|.35}$	120	300	187	5174	459	23027	29	0.035	0.289	0.062	0.036
	XGBOOST	240	300	1606	47307	7279	374169	436	0.033	0.181	0.056	0.034
Crypt XXX	$\text{TDA}_9^{.35|.35}$	90	300	77	2460	271	11057	14	0.030	0.221	0.053	0.031
	COSINE	30	600	589	20872	610	42952	65	0.027	0.491	0.052	0.028

Table V shows the main results of our models. For each ransomware family, TDA has a hyper-parameter set that produces the best model. However, these parameter values are not the same across families. Sample size (N) and training length (l) parameters are different as well. For each family, we also provide the best non-TDA model for comparison. The Locky ransomware has the best results with a precision of 0.161 and recall of 0.9. In general, models yield better recall than precision. In Table V, $\#w$ is the number of windows where a model makes at least one label prediction. By using the q, ϵ_1 and ϵ_2 hyper-parameters, TDA models avoid predicting labels when level of confidence in the derived classification is low.

Similar to TDA, DBSCAN can ignore data points in clustering; two of the best non-TDA results are delivered by DBSCAN models. In the best TDA models for each ransomware family, **we predict 16.59 false positives for each true positive**. In turn, this number is 27.44 for the best non-TDA models.

As TDA models appear to be sensitive to the selected hyper-parameters, we turn our attention to these values. For increasing threshold values, recall decreases from 0.53 ($q = 0.0$) to 0.37 ($q = 0.9$), but precision and PLR values do not change. In Figure 8, we present model performance for ϵ_1 (i.e., threshold on how many past ransomware addresses should be contained in the cluster) and ϵ_2 (i.e., threshold on at most how many addresses the cluster can contain). As Figure 8 indicates, TDA models tend to be more sensitive to ϵ_2 values. By limiting ϵ_2 to small values, we reach higher precision on average. However, this leads to fewer predictions, as ϵ_2 defines

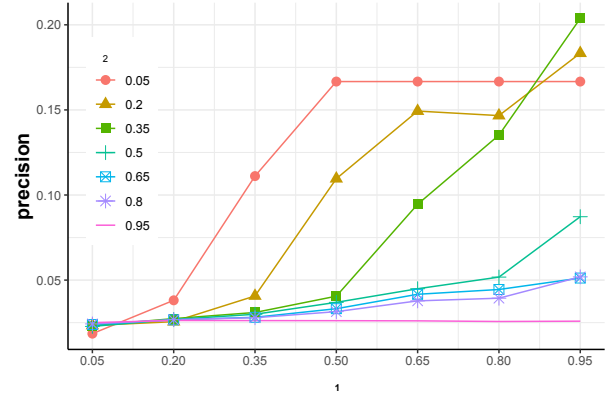Fig. 8: ϵ parameters in all TDA models (the filter threshold $q = 0.9$).

at most how many data points can be grouped together (see sec. IV-B).

Our results indicate that making predictions in every time window may not be desirable, as it leads to lower F1 values. Both TDA and DBSCAN can ignore some data points, thereby making predictions only when *enough* data exists. As we can control the sensitivity of TDA models through hyper-parameter selection, a natural question is to ask whether more restrictive settings should be preferred in TDA. Figure 9 depicts TDA performance when settings allow for lower (i.e., fewer windows) or higher (i.e., more windows) sensitivity. Precision, recall and F1 values are averaged for TDA models. We find that less sensitive models yield higher precision, but lower recall values. Cerber results show the greatest change in terms of sensitivity, whereas Locky has the best and most stable results. For some families, such as CryptXXX, the least sensitive models deliver the highest precision values. These findings imply that some TDA models are able to make predictions for very few windows only, but the results can be very accurate.

B. New Family Prediction

We outline our experimental settings for the detection problem as follows. Given features X_t and known labels Y_t of past addresses at time t and features of addresses $X_{t'}$ at time $t' > t$, we train for discovering addresses belonging to new ransomware families as follows:

1) For $t < t'$, use a training length l, and create a dataset $X_{[t-l,t]} \subseteq X_t$ which holds features of addresses observed between windows $t - l$ and t.
2) Create an f_0 sample of size N from $X^0_{[t-l,t]} \subseteq X_{[t-l,t]}$ without replacement where $\forall x_u \in X^0_{[t-l,t]}, y_u = f_0$ and $N = \left| X^0_{[t-l,t]} \right|$.
3) Create a *ransomware* sample of size N from $X^{rs}_{[t-l,t]} \subseteq X_{[t-l,t]}$ without replacement where $\forall x_u \in X^{rs}_{[t-l,t]}, y_u \neq f_0$ and $N \leq \left| X^{rs}_{[t-l,t]} \right|$.
4) Relabel all addresses in $Y^{rs}_{[t-l,t]}$ with the label f_r.
5) By using the ground truth data at t', take a sample of M=1000 white addresses without replacement to be used in
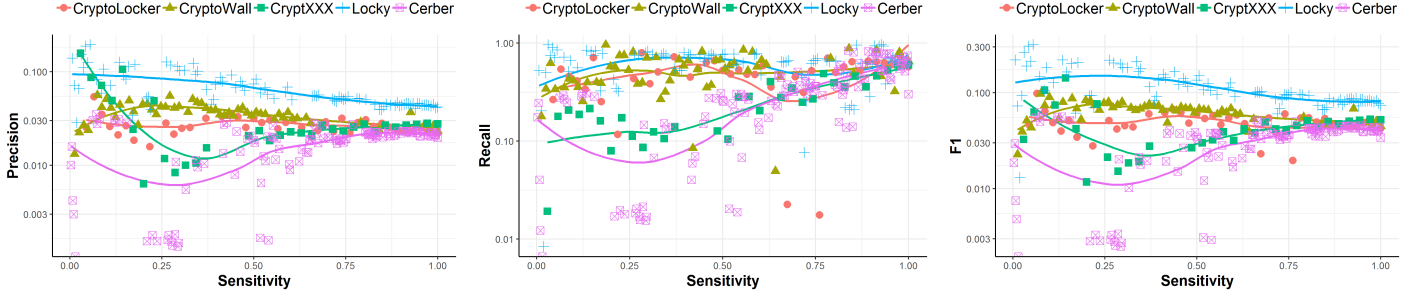
Fig. 9: Performance of TDA models with increasing sensitivity. Sensitive models can make predictions for every window.

the testing phase: $X_{t'}^0$.

6) By using the ground truth data, choose a family rs' whose emergence at t' will be discovered.

7) By using the ground truth data at t', find all ransomware addresses for t' to be used in the testing phase: $X_{t'}^{rs'}$.

8) Use features $\{X_{[t-l,t]}^0 \cup X_{[t-l,t]}^r\}$ and labels $\{Y_{[t-l,t]}^0 \cup Y_{[t-l,t]}^r\}$ as the training data, and classify $\{X_{t'}^0 \cup X_{t'}^r\}$.

We emphasize two aspects in predicting a new family: i) [Step 4]: in training, addresses of all existing families are relabeled with f_r, thereby creating a unified ransomware class, ii) [Step 8]: when an address is predicted as ransomware, we cannot immediately claim that it constitutes a new family, or belongs to an existing family.

As our current goal is to predict new ransomware families without any prior information about these families, *the training task has to learn a single model that will be used to identify all future ransomware families*. Using our models, we forecast emergence of 25 ransomware families. Emergence of the first ransomware, CryptoLocker, cannot be predicted, because we have no prior ransomware data to train a model for CryptoLocker.

The best model, $\mathbb{TDA}_{0.7}^{0.05|0.35}$ uses $N = 1000$ past samples, $l = 120$ training length, and predicts $TP = 26, FN = 8, TN = 5032, FP = 21075$. The model predicts 25 emerging ransomware families, but also **results in 810.57 false positives for each true positive**.

We hypothesize that this unsatisfactory performance is due to data scarcity; among the 25 families, only three families have more than one address in their first window on the Bitcoin blockchain. These families are DMALockerv3 (2016 day 233), Flyper (2016 day 335) and eRanger (2016 day 68).

TABLE VI: Day/Year pairs in the discovery experiment.

Ransomware	First window	Used window	#Unique address
Cerber	62/2016	89/2016	16
CryptXXX	132/2016	133/2016	38
DMALocker	7/2015	34/2016	14
CryptoWall	59/2014	64/2014	22
Locky	42/2016	47/2016	59

We now repeat the forecasting experiment by considering the *earliest* window when a ransomware yields 10 or more addresses. Such filtering results in a dataset of five ransomware families. We exclude the previously observed addresses of these families from our training set. The first and the identified

earliest windows for each family are presented in Table VI. Time difference in windows is as small as one for some families.

TABLE VII: Model performance for new family prediction.

Method	Training size N	Training length l	PLR
$\mathbb{TDA}_{0.9}^{0.2|0.2}$	300	60	0.21085470
COSINE	300	60	0.04232920
DBSCAN$_{0.2}$	300	60	0.02802958
RANDOMFOREST	300	60	0.00000000
XGBOOST	300	60	0.00000000

Despite improving data scarcity, Table VII shows that tree based methods (i.e., Random Forest and XGBoost) fail to predict any ransomware family. Table VIII shows predicted values for each family. We find that three TDA models tend to deliver the best F1 results for all five families. In addition to TDA models, we show one competing result from other models for each family. Overall, in three families TDA has the highest PLR value. For DMALocker, the best models are DBSCAN-based ones. Unlike existing family detection, in new family prediction we value finding an address with the fewest false positives. In this aspect, we reach the best result for CryptXXX, where a TDA model predicts one true positive and one false positive. With the best models provided in Table VIII, on average **we predict 27.53 false positives for each true positive** in forecasting of new ransomware families.

However, we emphasize that for some families, such as CryptXXX, our models predicts only two ransomware addresses, one of which is a true positive. This result offers evidence that our prediction models can be highly effective for certain families of ransomware.

Model parameters in prediction. Table IX shows average model performance across five ransomware families for different q values in TDA models. As the threshold increases, we observe slightly higher precision and lower recall values. Figures 10 and 11 depict PLR and recall values for each family. Overall, families exhibit better results for smaller training lengths (i.e., l) and larger sample size (i.e., N) values. This implies that ransomware addresses change features in time, and using a long history does not always result in better model performance.

11

RS	Method	Prec	Rec	TN	FP	TP	FN	PLR
CryptXXX	$\text{TDA}^{0.2|0.2}_{0.9}$	0.500	0.026	917	1	1	37	1.0
	COSINE	0.046	0.342	654	264	13	25	0.049
Locky	COSINE	0.098	0.138	795	37	4	25	0.108
	$\text{TDA}^{0.05|0.95}_{0.9}$	0.047	0.586	489	343	17	12	0.049
CryptoWall	$\text{TDA}^{0.05|0.95}_{0.9}$	0.0625	0.500	810	165	11	11	0.067
	$\text{TDA}^{0.35|0.8}_{0.9}$	0.061	0.500	805	170	11	11	0.0647
Cerber	$\text{TDA}^{0.05|0.95}_{0.9}$	0.029	0.214	849	100	3	11	0.030
	$\text{TDA}^{0.35|0.8}_{0.9}$	0.023	0.642	570	379	9	5	0.023
DMALocker	$\text{DBSCAN}_{0.2}$	0.019	0.875	120	367	7	1	0.019
	$\text{DBSCAN}_{0.15}$	0.015	0.875	4	459	7	1	0.015

TABLE IX: Mean values of metrics for the filter threshold in TDA models across five families.

q	PLR	Precision	Recall	TP
0.0	0.028	0.0264	0.664	15.25
0.3	0.028	0.0266	0.635	14.61
0.5	0.029	0.0278	0.589	13.75
0.7	0.031	0.0298	0.515	12.15
0.9	0.035	0.0319	0.352	8.35

Fig. 11: The impact of training size across best models of the five ransomware families.
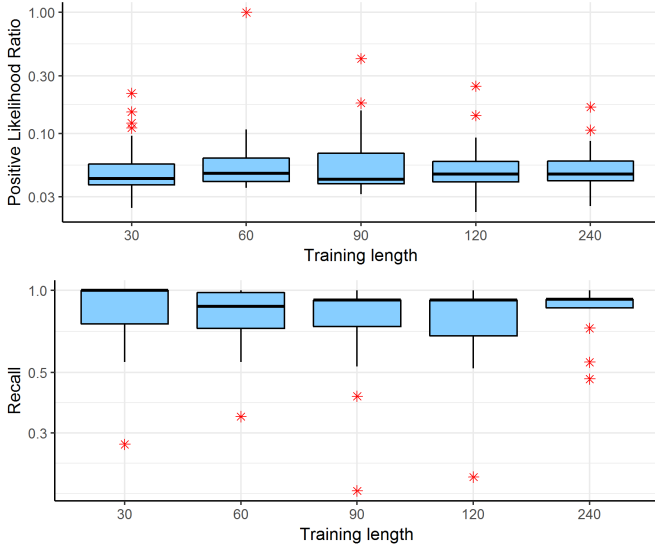

Fig. 10: Model parameters and performance values across best models of the five ransomware families.

VII. ADDRESS ELIMINATION

As a final step in our ransomware prediction, we employ three filtering approaches on the results produced by the best models in the existing family detection task.

Setting. To this end, we use the best TDA models outlined in Table V with $N = 300$, $l = 60$, and classify addresses in the last 30 days of each ransomware family. This task results in 72 days, where we can forecast and test the predictive performance. CryptoLocker, CryptXXX, DMALocker, Cerber and Locky have 2, 20, 1,19 and 20 days' predictions, respectively. In these forecasting tasks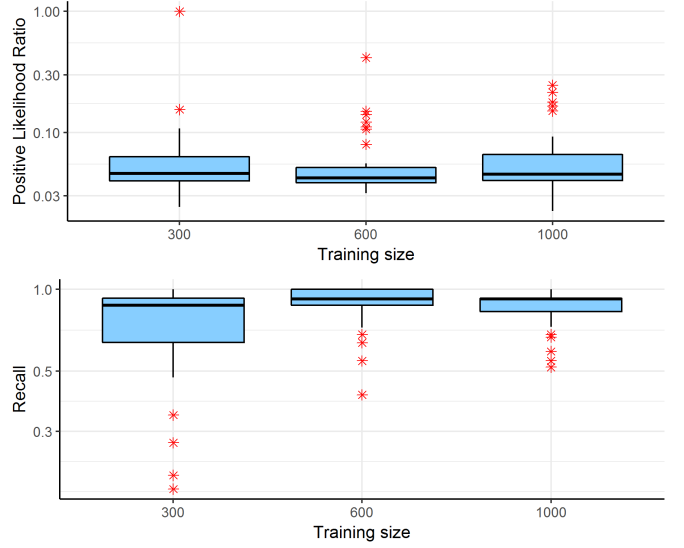, 3,211 addresses have been labeled as *suspicious* (potential ransomware), whereas only 92 of them are true ransomware addresses.

Shape Elimination. We hypothesize that ransomware transactions can be divided into two types: front and mixing. Front transactions are initial payments made by ransomed entities. Mixing transactions are created by hackers to transfer and sell received coins to other addresses. The ransomed entities buy bitcoins from exchanges or bitcoin users to pay hackers. These transactions are N-to-1 (i.e., in a transaction that has N input addresses and one output address, all coins are gathered in one address) or N-to-2 (i.e., a remaining amount is further directed to a change address) transactions. In this setting, depending on how many bitcoins are bought, N can be as high as 15000.

If we focus on detecting front payments only, we hypothesize that number of ransomware labeled addresses can be further reduced. On the complete Bitcoin blockchain, 91.06% of all transactions are N-to-1 or N-to-2 transactions. Of the 24198 known ransomware addresses in our dataset, 20826 (86.06%) have at least one appearance that receives coins in an N-to-1 or N-to-2 transactions. Of the 3,211 addresses, we find that 861 (26.8%) can be filtered out. However, 16 of the 92 true ransomware addresses are also filtered out. **Shape elimination results in 2,350 suspicious addresses and 76 ransomware addresses**.

Graph Elimination. Once ransom is received, hackers can sell each ransom coin to a different buyer (address) individually. This approach makes ransomware address detection particularly challenging. However, in most cases transaction costs discourage this privacy enhancing approach, and coins are merged and sold together in a single transaction. For example, Huang et al. [17] found that coins of Cerber family were continuously merged, and the resulting large tree of addresses was easily identified by the researchers.

We hypothesize that the small distance between an address to known past ransomware addresses can help us to be more
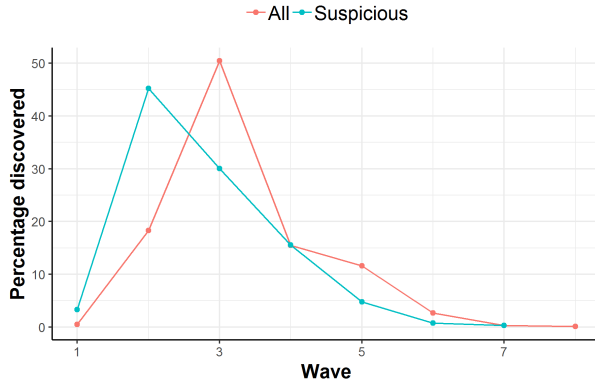
Fig. 12: Distance of ransomware labeled (suspicious) addresses and all addresses to known ransomware addresses. Suspicious addresses are closer to known ransomware. Difference of the two distributions is statistically significant as indicated by Kolmogorov-Smirnov test [20] with $D = 0.29564$, and p-value $< 2.2 \times 10^{-16}$.

confident in its label designation. We measure the distance in a breadth first search as the number of edges on the shortest path to a known ransomware address. Figure 12 depicts the distribution of 666 addresses which are reachable from the known past virus addresses on the graph. Here, 8 of the 92 true ransomware addresses are contained in these 666 addresses. Furthermore, the 8 unique addresses were at most 4 hops away from known ransomware addresses.

Figure 12 indicates that among all addresses that are reachable from the known ransomware addresses, the suspicious addresses tend to be located noticeably closer. Mean and median distance values of suspicious nodes to ransomware nodes are $\bar{x} = 2.44$, $\tilde{x} = 3$, respectively, with a maximum distance of 7 hops. If we use a maximum distance of 4 or less hops, only 617 addresses remain. **Graph elimination hence results in 617 suspicious addresses and 8 true ransomware addresses.**

Corroborating evidence: In all ransomware families, we record 324 addresses that are found to be suspicious in multiple time windows. In maximum, 3 addresses appear in 6 time windows each. All 324 addresses survive shape elimination, whereas only 158 addresses survive graph elimination. In total, we find **350 unique suspicious addresses and 8 true ransomware addresses after graph and shape eliminations**.

Multisig elimination. Bitcoin contains multiple types of addresses. In the simplest case, an *ordinary* address is controlled by a single private key. These addresses start with the character '1' such as 1LXrSb67EaH1LGc6d6kWHq8rgv4ZBQAcpU. A second address type, *multisig address*, requires more than one key to authorize a Bitcoin transaction. Multisig addresses start with the character '3' such as 3NQoq5MVPfEMw12gB4a2c1G61mRZyMymsB. These addresses are usually used to divide up responsibility for possession of bitcoins. For the first usage, as ransomware payments involve a payer and a payee who may not trust each other, we advocate that multisig addresses cannot be used in ransomware payments.

A second usage is to prevent coin theft: users can create

a multisig transaction where they own all the addresses, but private keys are stored on different machines. In such schemes, coins can only be stolen if all machines are compromised, which makes the theft very difficult. Hackers could potentially use multisig addresses to avoid theft of ransomed coins. Though, we believe that such a scenario is currently very rare, since only three addresses with multisig are found among the more than 24K known ransomware related addresses. Of course, this heuristic may not be effective in the future, if ransomware operators start using more multisig addresses.

Out of the 358 suspicious address, 339 are ordinary addresses; these addresses start with '1'. Hence, our multisig elimination round ends with 8 true ransomware, and 331 suspicious addresses.

Our elimination results show that many addresses are found suspicious in multiple time windows. Furthermore, such addresses survive all graph, chainlet and multisig eliminations. **Our results offer strong evidence that these suspicious addresses are indeed ransomware related.**

VIII. CONCLUSIONS

In this paper, we propose a new data analytics based framework to detect and predict ransomware transactions that use Bitcoin. Using a topological data analysis based approach and novel blockchain graph related features, we empirically show that we can significantly improve the ransomware related Bitcoin address detection accuracy compared to existing heuristic based approaches. Furthermore, we show that by using certain address elimination heuristics, we can reduce the number of suspicious addresses that need to be analyzed by the cyber security analysts.

As a future work, we plan to combine our proposed approach with other methods (e.g., running the ransomware to get additional seed addresses) and threat intelligence information (e.g., reports about the emergence of new ransomware) to increase our prediction accuracy.

REFERENCES

[1] C. G. Akcora, A. K. Dey, Y. R. Gel, and M. Kantarcioglu, "Forecasting bitcoin price with graph chainlets," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PaKDD)*. Springer, 2018, pp. 765–776.

[2] N. Andronio, S. Zanero, and F. Maggi, "Heldroid: Dissecting and detecting mobile ransomware," in *Research in Attacks, Intrusions, and Defenses*, H. Bos, F. Monrose, and G. Blanc, Eds. Cham: Springer International Publishing, 2015, pp. 382–404.

[3] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *IFCA*. Springer, 2013, pp. 34–51.

[4] D. F. Bauer, "Constructing confidence sets using rank statistics," *Journal of the American Statistical Association*, vol. 67, no. 339, pp. 687–690, 1972.

[5] G. Carlsson, "Topology and data," *Bul. of the AMS*, vol. 46, no. 2, 2009.

[6] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *The 22nd SIGKDD*. ACM, 2016, pp. 785–794.

[7] C. Cimpanu, "Georgia county pays a whopping $400,000 to get rid of a ransomware infection https://www.zdnet.com/article/georgia-county-pays-a-whopping-400000-to-get-rid-of-a-ransomware-infection/," *ZDNet*, Mar 2019.

[8] M. Conti, A. Gangwal, and S. Ruj, "On the economic significance of ransomware campaigns: A bitcoin transactions perspective," *Computers & Security*, 2018.

TABLE X: Symbols and their explanations.

Symbol	Explanation	Section
$\mathcal{G}, V, E, B$	Graph, its nodes, edges and node types	III-B
TX	set of all transactions	
$\phi(u)$	node type of u	
Γ_a^i and Γ_a^o	in and out neighbors of a	
$a_u, \vec{x}_u, y_u$	address, address feature vector, address label	IV
D	number of features	
f_o, f_i	non-rs address, address of ransomware family i	
$X_t, X_{t'}, Y_t$	training features, test features, training labels	
$tx_n, \mathcal{A}_o(n), A_u^o(n)$	transaction n, sum of its outputs, output of tx_n to address u	
a_u^t	appearance time t of address u	
I_u, L_u, W_u, C_u, O_u	income, length, weight, count, loop of address u	
$\mathcal{TX}$	starter transactions	
U, ξ	number of observed addresses, filter function	IV-C
$q, \epsilon_1, \epsilon_2$	quantile filter, inclusion threshold, size threshold	
$I, S, \{u_{jk}\}$	range,set of intervals, TDA clusters	
$\mathcal{CT}, C_c$	a cluster tree graph, a cluster in the graph	
A_c, V	addresses in cluster C_c, known past RS addresses in cluster C_c	
P, S	scores of addresses, set of suspicious addresses	
l, N, M	training lengths, sample size in training, sample size in testing	V
X_t^0, X_t^{rs}	at time t training features of non-RS addresses, rs addresses	
w	a specific time window	

TABLE XI: Mean feature values of ransomware families.

Ramsomware	Weight	Length	Looped	Income	Neighbor	Count	#address	# Unique address
APT	0.71	67.63	734.09	371987303	2.54	2047	11	2
Cerber	0.32	39.90	54.40	103224640	2.01	737.08	9223	9177
ComradeCircle	0.05	144.00	0	203320001	2	1241	1	1
CryptConsole	0.59	43.42	0	45463341	2	831.71	7	5
CryptoLocker	0.88	30.67	100.98	1840825184	2.88	308.32	9315	1509
CryptoTorLocker2015	1.19	20.58	166.63	680784614	10.21	220.36	55	32
CryptoWall	0.79	47.92	121.01	701610513	2	425.87	12390	1894
CryptXXX	0.37	47.44	61.02	135534326	2.01	791.84	2419	1354
DMALocker	0.97	38.82	430.48	889427618	1.85	897.16	251	21
DMALockerv3	0.54	37.71	94.55	610589614	1.16	962.88	354	143
EDA2	0.17	74.33	951.5	37852094	1.33	4355.16	6	4
Flyper	0.51	19.11	0.00	49777364	1.44	324.22	9	8
Globe	0.48	56.06	393.65	79555251	2.09	1623.90	32	7
GlobeImposter	0.51	26.47	54.87	1076278167	1.58	357.2	55	1
Globev3	0.45	71.82	151.44	119933550	2.12	1377.29	34	5
KeRanger	0.36	50.20	280.7	99990000	1	1021.1	10	10
Locky	0.37	46.90	88.65	244419979	1.25	1027.25	6625	6584
NoobCrypt	0.85	23.19	80.13	228913211	1.29	338.23	483	28
Razy	43.21	32.62	199.77	228205910311	10.23	744.85	13	1
Sam	0.06	6.00	1	2900000000	3	1	1	1
SamSam	0.73	44.67	123.15	1078297206	1.55	1143.63	62	44
VenusLocker	0.09	29.14	0	97142857	1.58	1	7	1
WannaCry	0.58	100.57	801.25	67141405	1.71	4422.36	28	5
XLocker	0.41	144.00	0	100000000	1	4511	1	1
XLockerv5.0	0.32	44.85	0	185710813	1.14	1063.57	7	3
XTPLocker	0.34	108.50	285.75	263743359	1.38	2938.88	8	3

[9] G. Di Battista, V. Di Donato, M. Patrignani, M. Pizzonia, V. Roselli, and R. Tamassia, "Bitconeview: visualization of flows in the bitcoin transaction graph," in *IEEE VizSec*, 2015, pp. 1–8.

[10] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," Naval Research Lab Washington DC, Tech. Rep., 2004.

[11] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[12] M. Fleder, M. S. Kester, and S. Pillai, "Bitcoin transaction graph analysis," *arXiv preprint arXiv:1502.01657*, 2015.

[13] E. Forgy, "Cluster analysis of multivariate data: efficiency versus interpretability of classifications," *Biometrics*, vol. 21, pp. 768–780, 1965.

[14] A. Greaves and B. Au, "Using the bitcoin transaction graph to predict the price of bitcoin," *No Data*, 2015.

[15] J. Hernandez-Castro, E. Boiten, and M. Barnoux, "The 2nd kent cyber security survey," *Kent University reports*, 2014, https://kar.kent.ac.uk/52891/.

[16] T. K. Ho, "Random decision forests," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, Aug 1995, pp. 278–282 vol.1.

[17] D. Y. Huang, D. McCoy, M. M. Aliapoulios, V. G. Li, L. Invernizzi, E. Bursztein, K. McRoberts, J. Levin, K. Levchenko, and A. C. Snoeren, "Tracking ransomware end-to-end," in *Tracking Ransomware End-to-end*. IEEE, 2018, pp. 1–12.

[18] J. Lewis, "Economic impact of cybercrime – no slowing down," *McAfee Report*, no. 2, 2018.

[19] K. Liao, Z. Zhao, A. Doupé, and G.-J. Ahn, "Behind closed doors: measurement and analysis of cryptolocker ransoms in bitcoin," in *2016*

APWG Symposium on Electronic Crime Research (eCrime). IEEE, 2016, pp. 1–13.

[20] H. W. Lilliefors, "On the kolmogorov-smirnov test for normality with mean and variance unknown," *Journal of the American statistical Association*, vol. 62, no. 318, pp. 399–402, 1967.

[21] P. Y. Lum, G. Singh, A. Lehman, T. Ishkanov, M. Vejdemo-Johansson, M. Alagappan, J. Carlsson, and G. Carlsson, "Extracting insights from the shape of complex data using topology," *Scientific reports*, vol. 3, p. 1236, 2013.

[22] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[23] A. Martin, J. Hernandez-Castro, and D. Camacho, "An in-depth study of the jisut family of android ransomware," *IEEE Access*, vol. 6, pp. 57 205–57 218, 2018.

[24] G. Maxwell, "Coinjoin: Bitcoin privacy for the real world," in *Post on Bitcoin Forum*, 2013.

[25] D. McGinn, D.and Birch, D. Akroyd, M. Molina-Solana, Y. Guo, and W. J. Knottenbelt, "Visualizing dynamic bitcoin transaction patterns," *Big data*, vol. 4, no. 2, pp. 109–119, 2016.

[26] S. Meiklejohn, M. Pomarole, G. Jordan, D. Levchenko, K.and McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: characterizing payments among men with no names," in *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 2013, pp. 127–140.

[27] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[28] A. Narayanan and M. Möser, "Obfuscation in bitcoin: Techniques and politics," *arXiv preprint arXiv:1706.05432*, 2017.

[29] M. Ober, S. Katzenbeisser, and K. Hamacher, "Structure and anonymity of the bitcoin transaction graph," *Future internet*, vol. 5, no. 2, pp. 237–250, 2013.

[30] M. Paquet-Clouston, B. Haslhofer, and B. Dupont, "Ransomware payments in the bitcoin ecosystem," *arXiv preprint arXiv:1804.04080*, 2018.

[31] C. Ramey, "The crypto crime wave is here," *Wall Street Journal*, no. April 28, 2018.

[32] D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," in *International Conference on Financial Cryptography and Data Security*. Springer, 2013, pp. 6–24.

[33] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "Coinshuffle: Practical decentralized coin mixing for bitcoin," in *European Symposium on Research in Computer Security*. Springer, 2014, pp. 345–364.

[34] N. Scaife, H. Carter, P. Traynor, and K. R. Butler, "Cryptolock (and drop it): stopping ransomware attacks on user data," in *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2016, pp. 303–312.

[35] G. Singh, F. Mémoli, and G. E. Carlsson, "Topological methods for the analysis of high dimensional data sets and 3d object recognition." in *SPBG*, 2007, pp. 91–100.

[36] M. Spagnuolo, F. Maggi, and S. Zanero, "Bitiodine: Extracting intelligence from the bitcoin network," in *International Conference on Financial Cryptography and Data Security*. Springer, 2014, pp. 457–468.

[37] M. Swan, *Blockchain: Blueprint for a new economy*. O'Reilly Media, Inc., 2015.

[38] Symantec, "Ransom.cryptowall," *Online*, 2019, www.symantec.com/security-center/writeup/2016-041912-5637-99.

[39] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.