Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)
Special Track on AI in FinTech

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

Cuneyt G. Akcora[1*], **Yitao Li**[2], **Yulia R. Gel**[3] and **Murat Kantarcioglu**[3]
[1]University of Manitoba, Canada
[2]Purdue University, USA
[3]University of Texas at Dallas, USA
cuneyt.akcora@umanitoba.ca, li3552@purdue.edu,{ygl,muratk}@utdallas.edu

Abstract

Recent proliferation of cryptocurrencies that allow for pseudo-anonymous transactions has resulted in a spike of various e-crime activities and, particularly, cryptocurrency payments in hacking attacks demanding ransom by encrypting sensitive user data. Currently, most hackers use Bitcoin for payments, and existing ransomware detection tools depend only on a couple of heuristics and/or tedious data gathering steps. By capitalizing on the recent advances in Topological Data Analysis, we propose a novel efficient and tractable framework to automatically predict new ransomware transactions in a ransomware family, given only limited records of past transactions. Moreover, our new methodology exhibits high utility to detect emergence of new ransomware families, that is, detecting ransomware with no past records of transactions.

1 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. The first application of Blockchain has been the Bitcoin cryptocurrency [Nakamoto, 2008]. Bitcoin's success has ushered an age known as the Blockchain 1.0 [Swan, 2015], and there are over 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 [Dingledine *et al.*, 2004]. This ease of usage and worldwide transaction availability of Bitcoin have been noticed by malicious actors. Pseudo-anonymity of cryptocurrencies has attracted the interest of a diverse body of criminals, transnational terrorist groups, and illicit users. Cryptocurrency related crime and criminal abuse of blockchain technologies are nowadays recognized as the fastest-growing type of cyber-crime [Lewis, 2018].

Using cryptocurrencies for ransomware payments appears to be substantially more prevalent than has been previously realized. As noted by Hernendex-Castro et al. [Hernandez-Castro *et al.*, 2014], among the respondents to their survey, "the prevalence of the CryptoLocker ransomware 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 ransomware payments and their overall economic impact is an emerging challenge of critical societal importance.

There have been efforts to analyze the cryptocurrency transactions using various heuristics. For example, the "co-spending" heuristic" is based on the idea that all input addresses of a transaction must belong to the same person since private keys associated with those accounts are needed to sign the transaction inputs [Meiklejohn *et al.*, 2016]). However, to our knowledge, none of the previous efforts has leveraged advanced data analytics based on topological and geometric concepts of the underlying blockchain transaction graph.

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

We can summarize the significance of our contributions:

- To the best of our knowledge, we are the first to introduce the machinery of topological and geometric data analytic tools not only to ransomware detection but to e-crime analysis on blockchain.
- We design six features that encode known Bitcoin transaction obfuscation patterns which exhibit high utility in predicting ransomware related activities.
- Using the ground truth data collected by various external studies, we show that the developed ransomware prediction approach based on the Topological Data Analysis (TDA) delivers substantially higher accuracy, compared to existing heuristic based and standard machine learning procedures.
- In addition to detecting new addresses associated with a known ransomware family, we show that our new methodology also exhibits high utility to detect the emergence of new ransomware families.

*Contact Author

2 Related Work

The success of Bitcoin [Nakamoto, 2008] has also encouraged significant usage of cryptocurrencies for illegal activities. The earliest results aimed at tracking the transaction network to locate bitcoins used in illegal activities, such as money laundering and blackmailing (e.g., [Androulaki *et al.*, 2013]), by using heuristics.

Bitcoin provides pseudo-anonymity; although all transactions are public by nature, user identification is not required to join the network. Mixing schemes (e.g., [Maxwell, 2013; Ruffing *et al.*, 2014]) exist to hide the flow of coins in the network. Earlier research results have shown that some Bitcoin payments can be traced [Meiklejohn *et al.*, 2016]. As a result, obfuscation efforts [Narayanan and Möser, 2017] by malicious users have become increasingly sophisticated.

In ransomware analysis, Montreal [Paquet-Clouston *et al.*, 2019], Princeton [Huang *et al.*, 2018] and Padua [Conti *et al.*, 2018] 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 have used decision rules on amounts and times of known ransomware transactions to locate undisclosed ransomware (CryptoLocker) payments [Liao *et al.*, 2016]. More recent studies are joint efforts between researchers and Blockchain analytics companies; Huang et al. [Huang *et al.*, 2018] identify shared hacker behavior and use heuristics to identify ransomware payments. The authors estimate that 20,000 victims have made ransomware payments. However, these studies do not extract features nor build machine learning models to detect ransomware payments and families.

Feature extraction has been studied for ransomware detection in the software security domain. In software code analysis, Cryptolock inspects ransomware programs and their activity for malicious characteristics [Scaife *et al.*, 2016]. In this line of work, studies on ransomware for mobile devices extract software code features to catch malicious programs (e.g., [Scaife *et al.*, 2016]). However, these approaches mainly target ransomware detection before ransomware infects a system, and do not consider Bitcoin transactions. A more recent approach of [Weber *et al.*, 2019] uses 166 custom designed features and applies graph convolutional networks to detect illicit transactions (not just ransomware). We report a performance comparison of our TDA-based tool regarding data given in [Weber *et al.*, 2019] in Section 6, and show that it provides better performance compared to the graph convolutional network based approach.

Similar TDA improvements are reported in the literature. For example, Abay et al. achieve better prediction accuracy with TDA based Persistent Homology tools [Abay *et al.*, 2019], and Li et al. achieve better anomaly detection accuracy with TDA based models [Li *et al.*, 2020]. However, these works are aimed at price or price anomaly prediction.

3 Background and Preliminaries

3.1 Ransomware

Ransomware is a 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. Besides computer systems, ransomware can also infect IoT and mobile devices [Martin *et al.*, 2018]. 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.

3.2 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 in TX (see [Akcora *et al.*, 2017] for a graph primer on Blockchains). 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 $\phi(v) = \{\textbf{Transaction}\}$. 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_a^i and Γ_a^o to refer to predecessors (in-neighbors) and successors (out-neighbors) of an address a, respectively.

4 Ransomware Detection and Prediction

In this paper, we state the following five questions to analyze ransomware behavior on the Bitcoin blockchain: ❶- Which features can we extract from the Bitcoin network to detect ransomware payments? ❷- Does a 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 Data Analytics companies? ❺- Based on the information about existing ransomware families at a 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.

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. Depending on a setting, y_u can designate a *white* (i.e., non-ransomware) address or a ransomware address. We associate timestamp t_u to represent the earliest time when the address a_u appeared in a blockchain transaction. An address can appear in Bitcoin multiple times. Let $f_1, \ldots, f_n$ be labels of known ransomware families which have been observed until time point t. We set f_0 to be the label of addresses which are **not** known to belong to any ransomware family and we assume them 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:

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 t, $\tilde{Y}_t \cap \{f_{rs}\} = \emptyset$, increase t such that $\tilde{Y}_t$ contains at least one f_{rs}). 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. 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 available set of features $X_{t'}$ and 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, $\forall y_j \in Y_t, y_j \neq f_{rs'}$. 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. 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 available set of features $X_{t'}$ and history (X_t, Y_t).

5 Methodology

To solve each of our two research problems, we use four existing (baseline) methods and propose a fifth solution based on Topological Data Analysis that achieves the best results. In this section, we will outline these methods. We start by discussing our features.

5.1 Graph Features

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. 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 two purposes. First, the induced 24 hour network allows us to capture how fast a coin moves in the network. The speed is measured by the number of blocks in the 24 hour window that contains a transaction involving the coin. Second, temporal information of transactions, such as the local time, has been found useful to cluster criminal transactions (see Figure 7 in [Huang *et al.*, 2018]).

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: $I_u = \sum_{t_n \in \Gamma^o_u} A^o_u(n)$.

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

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 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., $\mathbb{TX} = \{\forall tx_n \in TX, s.t., \Gamma^i_n = \{a^{t^0}_1, \ldots, a^{t^n}_z\}, t^0 \leq t^n < t\}$. 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 the fraction of coins that originate from a starter transaction and reach u. Each output address u of a transaction tx_n receives $1/\Gamma^o_n$ coins, regardless of the amount $A^o_u(n)$. Note that *weight is oblivious to the transacted amount*. This design makes the weight feature robust against obfuscation that uses big coin flows to many other addresses.

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: We designed graph features to quantify specific obfuscation patterns used by ransomware operators:

Loop counts how many transactions 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 [McGinn *et al.*, 2016] for examples of such patterns).

Weight quantifies the merge behavior (i.e., the transaction has more input 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 [Huang *et al.*, 2018] for an application of this pattern).

Similar to weight, we design the count feature 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 quantifies mixing rounds [Maxwell, 2013] 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 [Ruffing *et al.*, 2014]).

5.2 Baseline Methods for Ransomware Prediction

Naive Similarity Search (1): We use addresses in a specific time window t and compute pairwise Cosine similarity [Karypis *et al.*, 2000] to known ransomware addresses from past l days.

Heuristics: The following heuristics [Meiklejohn *et al.*, 2016] are used in our experimental evaluation. **Co-spending heuristic**

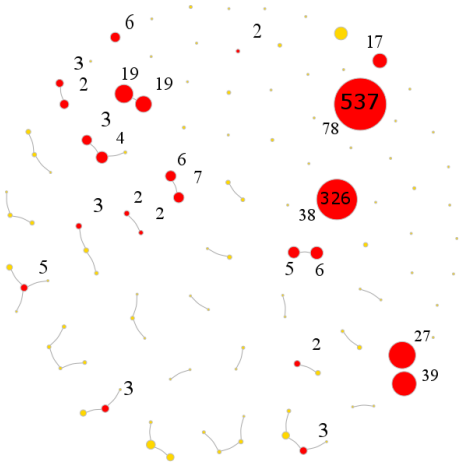

Figure 1: Mapper graph of the Cerber ransomware addresses (2017 day 307), filtered with the length attribute. Around clusters we indicate the number of past ransomware addresses contained in the cluster. We show the total order (i.e., address count) of two biggest clusters inside circles (326 and 537). The two clusters with 19 past ransomware addresses (top left) contain 53 and 67 addresses. Clusters without past ransomware are depicted in yellow.

(2A): "If two addresses are inputs to the same transaction, the same user controls them". **Transition heuristic (2B):** "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".

Clustering - DBSCAN (3A): DBSCAN is a density-based non-parametric clustering algorithm. DBSCAN can mark outlier points that lie alone in low-density regions as noise [Ester *et al.*, 1996].

Clustering - Hierarchical (3B): We use k-means clustering with Forgy based initial seed selection on address feature vectors [Forgy, 1965].

Extreme Gradient Boosting Trees (4A): XGBT applies gradient boosting algorithms to decision trees [Chen and Guestrin, 2016].

Random Forest (4B) is a supervised ensemble of multiple simple decision trees [Ho, 1995].

5.3 TDAMapper for Ransomware Analysis

We now introduce the concepts of Topological Data Analysis (TDA) into detection of ransomware patterns on Bitcoin. 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 [Carlsson, 2009].

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

As a result, Mapper produces a low dimensional representation of the underlying data structure in the form of a "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 Figure 1, we show an example of the produced Mapper graph. Each node may contain three sets of addresses: past RS addresses, past non-RS addresses, and addresses of the current time window, whose labels are unknown. If current addresses are contained in clusters that also contain many past known ransomware addresses, by association, we deem these current addresses potential ransomware addresses.

Algorithm 1 TDA filtering with multiple attributes.

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.
for 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**
      for $a_u \in A_c \setminus \{RS \cup NRS\}$ **do**
        $P_u \leftarrow 1 + P_u$
$q_t \leftarrow quantile(P, q)$
return $\{\forall a_u \in P | P_u \geq q_t\}$

We filter the TDA mapper graph by using each of our six graph features. As a result, we get 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 (see Algorithm 1).

Algorithm 1 starts by computing the number of past 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. 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 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.

Main idea: By observing the intrinsic topology and geometry of the blockchain transaction graph, Mapper allows for recovering of hidden similarities between "clusters", or groups of addresses,

that are typically unavailable with traditional clustering techniques. If an address appears in attribute filtered clusters with known ransomware addresses frequently, it is more likely to be a ransomware address itself.

6 Ransomware Detection and Prediction

Dataset. We have downloaded and parsed the entire Bitcoin transaction graph since its beginning in 2009. Using a time interval of 24 hours, we have extracted daily transactions on the network and formed the Bitcoin graph. For computational efficiency, we have filtered out the network edges that transfer less than ฿0.3, since ransom amounts are rarely below this threshold.

Metrics and Parameters. We compute accuracy by using overall sums of TN, FN, FP and TP values across multiple time windows as $(TP+FN)/(P+N)$. In all models, we report the optimal parameters that maximize F1 scores in predictions: 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 and interval = 80.

6.1 Existing Family New Address Detection

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:

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) from the test dataset we remove appearances of addresses that have appeared in the past (i.e., $t < t'$), since we already know their labels, ii) if an address appears in multiple windows, its each appearance has (potentially) different features in $X_{[t-l,t]}$ with the same rs label, iii) on many days, we do not have N past rs addresses to train from, iv) Most importantly, **we learn a model for each ransomware family**. In our analysis, we show that these models do not share the same characteristics.

Heuristics. Using co-spending and transition heuristics with all 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.

Authors of the three datasets that we adopt had already considered heuristics. Since the creation of these datasets, there have been very few addresses that are involved in transactions by past ransomware addresses.

RS	Method	l	#w	N	TP	TN	Acc.	Gain (%)
Crypto Locker	$\mathbb{TDA}^{.65\|.65}_{.9}$	240	300	34	439	22K	0.69	213.8
	$\mathbb{DBSCAN}_{.15}$	60	300	67	935	11K	0.22	
Crypto Wall	$\mathbb{TDA}^{.8\|.65}_{.9}$	240	600	15	217	11200	0.77	65.0
	$\mathbb{DBSCAN}_{.2}$	240	600	59	728	16913	0.47	
Crypt XXX	$\mathbb{TDA}^{.35\|.35}_{.9}$	90	300	14	77	11K	0.80	19.9
	$\mathbb{COSINE}$	30	600	65	589	42K	0.69	
Locky	$\mathbb{TDA}^{.8\|.5}_{.9}$	240	300	11	451	8221	0.78	2.9
	$\mathbb{COSINE}$	90	300	194	2395	146K	0.76	
Cerber	$\mathbb{TDA}^{.5\|.35}_{.9}$	120	300	29	187	23K	0.80	-7.8
	$\mathbb{XGBOOST}$	240	300	436	1.6K	374K	0.87	

Table 1: Task 1:Existing family undisclosed address detection.

Table 1 shows the main results of our models. Gain is defined as improvement of accuracy over the next best model (i.e., $Gain = 100 \times (Acc_{TDA} - Acc_{base})/Acc_{base}$); in CryptoLocker and CryptoWall this model is DBSCAN. Naive Cosine similarity search is the best baseline model in Locky and CryptXXX families. For each ransomware family, TDA has a hyper-parameter set that produces the best model. These parameter values are not the same across families. Sample size (N) and training length (l) parameters are different. For each family, we also provide the best non-TDA model for comparison. The CryptoLocker ransomware has the best TDA gain result with an improved accuracy of 213.8%. In Table 1, $\#w$ 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 the level of confidence in the derived classification is low.

Similar to TDA, DBSCAN can ignore data points in clustering, and DBSCAN yields two of the best non-TDA results. In the best TDA models for each ransomware family, **we predict 16.59 FP for each TP (lower is better)**. This number is 27.44 for the best non-TDA models.

Other approaches. The Elliptic dataset study [Weber *et al.*, 2019] detects illicit transactions by using a Graph Convolutional Network, but Random Forests (RF) achieves higher accuracy. The Elliptic dataset does not disclose how each feature is generated nor provides the actual transactions or addresses used in transactions. Hence, we could not generate the features used in our study to compare utility of the features we have proposed. However, similar to our approach, the authors have divided the network into (50) time periods, and have used the latest periods as the test data. We applied our TDA approach to the Elliptic dataset by using the 166 features found in [Weber *et al.*, 2019] to detect *illicit* transactions and to understand the relative performance of our TDA tool. We have found that our TDA-based approach has detected 322 TP vs. 362 in RF in windows 40, 41 and 42, but correctly detected up to six times more TP transactions (14 vs 131 tp) in periods $43, \ldots, 50$. In particular, in four of the last seven windows TDA

RS	First	Used	#Unique add.
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

Table 2: Day/Year pairs in the discovery experiment.

can detect illicit transactions, but RF cannot detect any.

6.2 New Family Prediction

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 now train for discovering addresses belonging to **new ransomware families**:

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

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: $X^{rs'}_{t'}$.

8. Use features $\{X^0_{[t-l,t]} \cup X^r_{[t-l,t]}\}$ and labels $\{Y^0_{[t-l,t]} \cup Y^r_{[t-l,t]}\}$ as the training data, and classify $\{X^0_{t'} \cup X^r_{t'}\}$.

We emphasize two aspects in predicting a new family: i) in training, addresses of all existing families are relabeled with f_r, creating a unified ransomware class, ii) when an address is predicted as ransomware, we cannot immediately claim whether it makes up a new family or belongs to an existing family. As our current goal is to predict new ransomware families with no prior information about these families, the training task has to **learn a single model that will identify all future ransomware families**. Using our models, we forecast the emergence of 25 ransomware families. Emergence of the first ransomware, CryptoLocker, cannot be predicted since we have no prior data to train a model.

The best model, $\mathbb{TDA}^{0.05|0.35}_{0.7}$ uses $N = 1K$ 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 FP for each TP**. We hypothesize that this performance is because of data scarcity in training; 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 (2016/335) and eRanger (2016/68).

We repeat the forecasting experiment by considering the *earliest* window when a ransomware uses ten or more addresses. Such filtering results in a dataset of five RS 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 2. Time difference in windows is as small as one day for some families.

RS	Method	TN	TP	Acc.	Gain (%)
Cerber	$\mathbb{TDA}^{0.05\|0.95}_{0.9}$	849	3	0.88	47.1
	$\mathbb{TDA}^{0.35\|0.8}_{0.9}$	570	9	0.60	
CryptXXX	$\mathbb{TDA}^{0.2\|0.2}_{0.9}$	917	1	0.96	37.6
	$\mathbb{COSINE}$	654	13	0.69	
CryptoWall	$\mathbb{TDA}^{0.05\|0.95}_{0.9}$	810	11	0.82	0.61
	$\mathbb{TDA}^{0.35\|0.8}_{0.9}$	805	11	0.81	
Locky	$\mathbb{TDA}^{0.05\|0.95}_{0.9}$	489	17	0.58	-57.9
	$\mathbb{COSINE}$	795	4	0.92	
DMALocker	$\mathbb{DBSCAN}_{0.2}$	120	7	0.25	998
	$\mathbb{DBSCAN}_{0.15}$	4	7	0.02	

Table 3: Task 2: Detecting new RS family ($l = 60$, $N = 300$).

Table 3 shows the main results of our models. We define gain as improvement of accuracy over the next best model (i.e., $Gain = 100 \times (Acc_{TDA} - Acc_{base})/Acc_{base}$). We find that three variations of TDA models deliver the best F1 results for all five families. Besides TDA models, we show one competing result from other models for each family. Overall, in three families TDA has the highest accuracy value. We reach the best result for CryptXXX, where a TDA model predicts 1 TP and 1 FP. With the best models provided in Table 3, on average **we predict 27.53 FP for each TP** in forecasting of new ransomware families.

We emphasize that for some families, such as CryptXXX, our models predict only two ransomware addresses, one of which is a TP. This result offers evidence that our prediction models tend to be highly effective for certain families of ransomware. These results also show that we can detect addresses from an emerging RS by using information from previously detected ransomware addresses. Finally, such findings imply that RS operators tend to behave similarly, which may indicate a shared origin between RS families.

7 Conclusions

We have proposed a new framework to detect and predict ransomware payments on Bitcoin using the advanced data analytic machinery of Topological Data Analysis. The new TDA-based tool has substantially improved the ransomware detection accuracy, compared to existing approaches.

As a future research direction, we plan to extend the proposed topological and geometric methodology to mapping entities to IP addresses in geographical locations.

Acknowledgements

Akcora has been partially supported by the NSERC Discovery Grant RGPIN-2020-05665. Gel has been partially supported by NSF DMS 1925346, IIS 1633331, NSF DMS-1736368 and ECCS 1824716. Kantarcioglu has been partially supported by NIH 1R01HG006844, NSF CICI-1547324, and IIS-1633331.

References

[Abay *et al.*, 2019] N. C. Abay, C.G. Akcora, Y.R. Gel, U.D. Islambekov, M. Kantarcioglu, and B. Thuraisingham. Chainnet: Learning on blockchain graphs with topological features. In *IEEE ICDM*, 2019.

[Akcora *et al.*, 2017] C. G. Akcora, Y. R. Gel, and M. Kantarcioglu. Blockchain: A graph primer. *arXiv preprint arXiv:1708.08749*, pages 1–17, 2017.

[Androulaki *et al.*, 2013] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun. Evaluating user privacy in bitcoin. In *IFCA*, pages 34–51, 2013.

[Carlsson, 2009] G. Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2), 2009.

[Chen and Guestrin, 2016] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *22nd ACM SIGKDD*, pages 785–794, 2016.

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

[Dingledine *et al.*, 2004] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab, 2004.

[Ester *et al.*, 1996] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *ACM SIGKDD*, volume 96, pages 226–231, 1996.

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

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

[Ho, 1995] Tin Kam Ho. Random decision forests. In *3rd ICDAR*, volume 1, pages 278–282 vol.1, 1995.

[Huang *et al.*, 2018] 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*, pages 1–12. IEEE, 2018.

[Karypis *et al.*, 2000] M. Karypis, G. Steinbach, V. Kumar, and M. Steinbach. A comparison of document clustering techniques. In *KDD Workshop on TM*, volume 400, pages 525–526, 2000.

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

[Li *et al.*, 2020] Y. Li, U. Islambekov, C. Akcora, E. Smirnova, Y. R. Gel, and M. Kantarcioglu. Dissecting ethereum blockchain analytics: What we learn from topology and geometry of the ethereum graph? In *SIAM SDM*, pages 523–531, 2020.

[Liao *et al.*, 2016] K. Liao, Z. Zhao, A. Doupé, and G.-J. Ahn. Behind closed doors: measurement and analysis of cryptolocker ransoms in bitcoin. In *IEEE APWG Symposium on eCrime*, pages 1–13, 2016.

[Martin *et al.*, 2018] A. Martin, J. Hernandez-Castro, and D. Camacho. An in-depth study of the jisut family of android ransomware. *IEEE Access*, 6:57205–57218, 2018.

[Maxwell, 2013] Greg Maxwell. Coinjoin: Bitcoin privacy for the real world. In *Post on Bitcoin Forum*, 2013.

[McGinn *et al.*, 2016] D. McGinn, D.and Birch, D. Akroyd, M. Molina-Solana, Y. Guo, and W. J. Knottenbelt. Visualizing dynamic bitcoin transaction patterns. *Big data*, 4(2):109–119, 2016.

[Meiklejohn *et al.*, 2016] 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 *ACM IMC*, pages 127–140, 2016.

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

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

[Paquet-Clouston *et al.*, 2019] M. Paquet-Clouston, B. Haslhofer, and B. Dupont. Ransomware payments in the bitcoin ecosystem. *Journal of Cybersecurity*, 5(1):tyz003, 2019.

[Ruffing *et al.*, 2014] T. Ruffing, P. Moreno-Sanchez, and A. Kate. Coinshuffle: Practical decentralized coin mixing for bitcoin. In *ESORICS*, pages 345–364, 2014.

[Scaife *et al.*, 2016] N. Scaife, H. Carter, P. Traynor, and K. R.B. Butler. Cryptolock (and drop it): stopping ransomware attacks on user data. In *IEEE 36th ICDCS*, pages 303–312, 2016.

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

[Weber *et al.*, 2019] M. Weber, G. Domeniconi, J. Chen, D. K. I. Weidele, C. Bellei, Tom Robinson, and C. E. Leiserson. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv:1908.02591*, 2019.