Supplementary Material for Chartalist: Labeled Graph Datasets for UTXO and Account-based Blockchains

1 Ransomware Dataset

1.1 BitcoinHeist features

On the heterogeneous Bitcoin network, in-neighbors Γ_n^i 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. Out-neighbors of tx_n are denoted as Γ_n^o. A transaction has inputs and outputs; the sum of output amounts of a transaction tx_n is defined as $\mathcal{A}^o(n) = \sum_{a_u \in \Gamma_n^o} A_u^o(n)$, where an output address a_u receives $A_u^o(n)$ 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_u^t. To mine address behavior in time, we divide the Bitcoin network into 24-hour long windows by using the UTC-6 timezone as a 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 number of blocks measures the speed in the 24-hour window that contains a transaction involving the coin. Second, temporal information of transactions, such as the local time, has been useful to cluster criminal transactions.

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

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

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 the 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., $\mathbb{TX} = \{\forall tx_n \in TX, s.t., \Gamma_n^i = \{a_1^{t^0}, \ldots, a_z^{t^n}\}, t^0 \leq t^n < t\}$. 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 $1/\Gamma_n^o$ coins, regardless of the amount $A_u^o(n)$. 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 that 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 that 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.

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.

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 on Bitcoin, where transactions receive and distribute similar amounts of coins in multiple rounds with newly created addresses to hide the coin origin.

1.2 Ransomware Families

We report the counts and feature averages of the ransomware families in Table 1.

Table 1: Mean feature values of ransomware families.

Ransomware	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

2 Ethereum Address Classification Dataset

AlphaCore relies on a single computed property, namely the data depth of multiple node properties, to determine core membership. During AlphaCore execution, these data depth values are iteratively updated by computing node property functions and applying data depth on the resulting values. In our baseline models, we use the features of an address given in Table 2.

3 Dataset Structures

3.1 Bitcoin Transaction Network

The Bitcoin network is growing very fast. The number of transactions has risen considerably between 2011 and 2022. As a result, the number of transactions in each block is substantially higher than early days of the network. figure 1 shows the number of unique addresses and transactions for the bitcoin network over time.

This dataset contains transaction edge files from block 0 to block 737900. Block times are given in the bitcoin_times.csv file (US Central time). The data is divided into ten splits. Each split's data is zipped together and contains 73600 block edge files of transactions that were mined in the blocks of that splitting scope. Each .csv file contains the information for 100 blocks. Each line in the input edge file is tab-separated with the format:

```
blockNo \t hash of transaction \t numOfInputs \t first input address \t first input amount \t second input address \t
second input amount \t (additional inputs, if exist) \r\n
```

Each line in the output edge file is tab-separated with the format:

```
blockNo \t hash of transaction \t numOfInputs \t first output address \t first output amount \t second output address \t
second output amount \t (additional outputs, if exist) \r\n
```

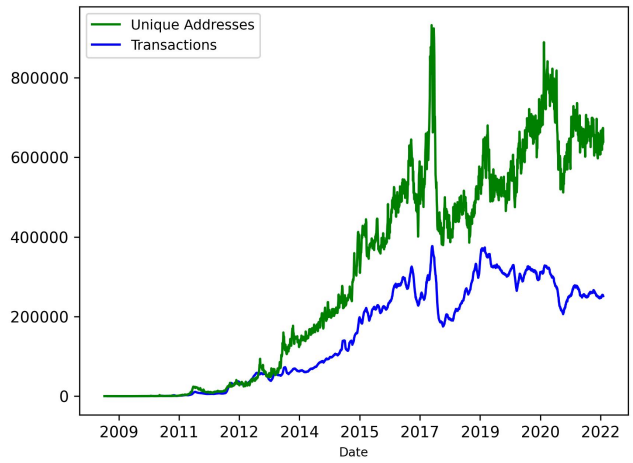

Figure 1: **Number of unique addresses and transactions for bitcoin network**. The number of unique addresses (i.e., vertices in the transaction graph) has increased above 600K.

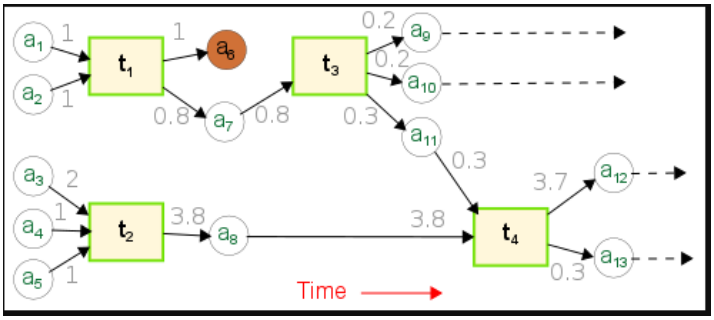

Figure 2: **Sample bitcoin transaction graph.** This figure shows a transaction graph with four transaction.

Table 2: **Example node property functions.** Most functions are adopted from the related work. Functions that encode the community around a node, such as cycles, can help bringing a higher ordered structure of the network into use.

Function	Value / number of ...
$N(u)$	neighbors of u
$N_{out}(u)$	neighbors reachable with outgoing edges from u
$N_{in}(u)$	neighbors reachable with incoming edges to u
$deg(u)$	edges to/from u (Degree)
$deg_{out}(u)$	outgoing edges from u (Out-Degree)
$deg_{in}(u)$	incoming edges to u (In-Degree)
$\bigcirc(u,l)$	undirected cycles of length l that u is part of
$\circlearrowright(u,l)$	directed cycles of length l that u is part of
$t(u,l)$	length l timeframes that u has edges in
$S(u)$	sum of edge weights incident to a node (Strength)
$S_{out}(u)$	sum of outgoing edge weights (Out-Strength)
$S_{in}(u)$	sum of incoming edge weights (In-Strength)

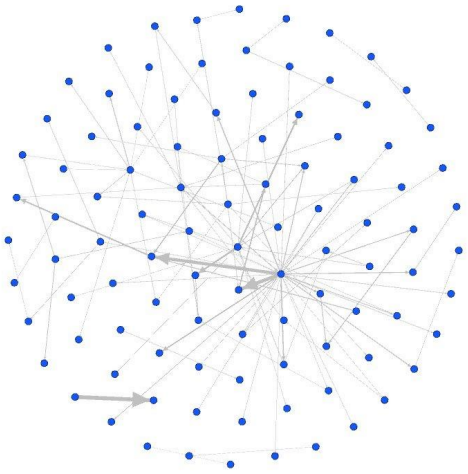

Figure 3: **Bytom token transaction network.** Edges are aggregated to indicate the number of transactions between addresses.

Consider the Bitcoin graph in the figure 2, where transactions are shown with rectangles and addresses are shown with circles. There are four transactions and 13 addresses. This graph would be given in and out-edge files. Below, we show the content of each file. Note that we list previous (unknown) transactions with t_x* notation (For example, t_x1, t_x2 are not shown in the network figure because they have appeared in the earlier blocks).

In edges of transactions

```
BlockHeightOft_1 HashOft_1 2 a1 1 a2 1
BlockHeightOft_2 HashOft_2 3 a3 2 a4 1 a5 1
BlockHeightOft_3 HashOft_3 1 a7 0.8
BlockHeightOft_4 HashOft_4 2 a11 0.3 a8 3.8
```

Out edges of transactions

```
BlockHeightOft_1 HashOft_1 2 a_6 10^8 a_7 0.8^0.8
BlockHeightOft_2 HashOft_2 1 a_8 3.8*10^8
BlockHeightOft_3 HashOft_3 3 a_9 0.2*10^8 a_10 0.2*10^8 a_11 0.3*10^8
BlockHeightOft_4 HashOft_4 2 a_12 3.7*10^8 a_13 0.3*10^8
```

3.2 Ethereum Networks

On account blockchains, we can observe the following networks:

1. Coin transaction network: Similar to the UTXO transaction network, this network is created from the coin (ether) transfers between addresses. Network edges only carry the native currency (coin) of the blockchain.

2. Token transaction networks: crypto asset trading networks that are created by internal smart contract transactions.

Table 3: Forecasting performance (MAPE in %) on Ethereum networks

Model	Bytom	Decentraland	Golem
FC-LSTM [5]	40.72	33.46	35.73
DCRNN [4]	35.36	27.69	23.15
STGCN [7]	37.33	28.22	23.68
GraphWaveNet [6]	39.18	37.67	28.89
AGCRN [1]	34.46	26.75	22.83
Z-GCNETs [3]	31.04	23.81	22.32
StemGNN [2]	34.91	28.37	22.50
TAMP-S2GCNets (us)	**29.26**	**19.89**	**20.10**

3. Trace network: interactions between all address types. The name trace implies that a transaction triggers a cascade of calls to smart contracts or externally owned addresses.

Currently, Chartalist provides token network datasets.

3.2.1 Token Transaction Network

A token transaction network has EOA, NULL, and smart contract addresses as nodes. We outline the following three types of transactions that a Data Scientist must know to analyze token networks.

1. The creation transaction that assigns an address for the token, initializes its smart contract and state variables.
2. A trade transaction that moves some tokens between addresses.
3. A management transaction that can only be initiated by the smart contract creator (or any address that the owner specifies). The transaction may delete the contract, or forward its balance (in ether or token) to another address.

Token network datasets has the following format:

```
(tokenID) \t fromAddress \t toAddress \t unixtime \t Amount.
```

Figure 3 shows the entire Bytom network where edges indicate the number of transactions between addresses. For visual clarity, all address types are shown with the same circular shape.

4 Leaderboards

The articles that we have cited for each dataset have extensive comparisons with existing approaches, however Blockchain Data Analytics is still a nascent field and we do not have many results from other researchers on our tasks.

The Ethereum price prediction task has a current leaderboard with the following metrics reported in Table 3.

References

[1] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in Neural Information Processing Systems*, 33, 2020.

[2] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. Spectral temporal graph neural network for multivariate time-series forecasting. In *Advances in Neural Information Processing Systems*, volume 33, pages 17766–17778, 2020.

[3] Yuzhou Chen, Ignacio Segovia-Dominguez, and Yulia R. Gel. Z-gcnets: Time zigzags at graph convolutional networks for time series forecasting. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 1684–1694. PMLR, 2021.

[4] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.

[5] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

[6] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019.

[7] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3634–3640, 2018.