Shoreline: Data-Driven Threshold Estimation of Online Reserves of Cryptocurrency Trading Platforms

Xitong Zhang
Michigan State University
zhangxit@msu.edu

He Zhu
OceanEx Labs, BitOcean Global
hezhu@bitocean.org

Jiayu Zhou
Michigan State University
jiayuz@msu.edu

Abstract

With the proliferation of blockchain projects and applications, cryptocurrency exchanges, which provides exchange services among different types of cryptocurrencies, become pivotal platforms that allow customers to trade digital assets on different blockchains. Because of the anonymity and trustlessness nature of cryptocurrency, one major challenge of crypto-exchanges is asset safety, and all-time amount hacked from crypto-exchanges until 2018 is over $1.5 billion even with carefully maintained secure trading systems. The most critical vulnerability of crypto-exchanges is from the so-called *hot wallet*, which is used to store a certain portion of the total asset of an exchange and programmatically sign transactions when a withdraw happens. Whenever hackers managed to gain control over the computing infrastructure of the exchange, they usually immediately obtain all the assets in the hot wallet. It is important to develop network security mechanisms. However, the fact is that there is no guarantee that the system can defend all attacks. Thus, accurately controlling the available assets in the hot wallets becomes the key to minimize the risk of running an exchange. However, determining such optimal threshold remains a challenging task because of the complicated dynamics inside exchanges. In this paper, we propose SHORELINE, a deep learning-based threshold estimation framework that estimates the optimal threshold of hot wallets from historical wallet activities and dynamic trading networks. We conduct extensive empirical studies on the real trading data from a trading platform and demonstrate the effectiveness of the proposed approach.

Introduction

Ever since the introduction of the blockchain design in early 2013 (Nakamoto and others 2008), the technology has attracted significant interests from various companies and is transforming many industries. Notably, its *immutability* and tamper detection provide a revolutionary infrastructure for the supply chain management that provides trustless verification among different parties. Also, it leads to the revolution of finance that provides real-time and low-cost transactions built upon its decentralized governance and trustless transactions. Facebook's recent payment oriented blockchain system Libra (Facebook 2019) further paves the way to mass adoption of blockchain in individual users.

The economic foundation of blockchain projects is the cryptocurrencies, which are digital assets being transacted on the blockchains. The market value of the cryptocurrency principally reflects the economic value of a blockchain. At cryptocurrency exchanges, those who possess cryptocurrencies can use the currency in blockchains to exchange other digital assets and 'cash out' fiat. As such, exchanges are in the center of the blockchain economy and enable the liquidity of cryptocurrencies to flow among different parties.

Similar to traditional exchanges, such as stock brokerage exchanges and foreign exchange markets, a blockchain exchange manages accounts to accept deposits from, and issue withdraws to its customers, builds and maintains order books for the customers to trade at specific prices. However, different from traditional financial systems where fiats transactions are being settled between real persons, each blockchain transaction is between two addresses and is considered finalized once the sender signs the transaction by the corresponding private key. Such anonymity has brought significant security risks to crypto exchanges. For example, if a hacker managed to control a compromised server of the crypto exchange and steals the private key, the hacker can immediately withdraw all the assets in the exchange by signing a transaction to one of his/her addresses using the stolen private key.

The private key that is stored in the exchange server and programmatically signs transactions when a customer requests a withdraw is referred to as a *hot wallet*. The hot wallet has been the most critical vulnerability of crypto-exchanges, and it has been reported that all-time amount hacked from crypto-exchanges until 2018 is over $1.5 billion (Larcheveque 2018). Technically, the private key is a random number, and the signing is a deterministic algorithm, which means signing a transaction can be done completed in a disconnected computer, and the signed transaction can be broadcasted to the blockchain nodes in another computer. This offline signing technique is called cold storage, and the offline private keys are usually called *cold wallets*.

Even though the cold wallets can be almost impossible to hack, the hot wallet cannot be entirely replaced by cold

wallets: 1) withdraw from cold storage has substantial operational costs, the programmatic withdrawal from hot wallets ensures efficient operations, 2) signing from cold storage increases the exposure of private keys and thus increases systematic risk of getting leaked, e.g., from social engineering. *Thus, one of the popular design of exchanges is to reserve part of assets in the hot wallet for high-frequency trading considering operational efficiency to avoid frequent time-consuming refilling and store the rest in the cold wallet for security. When the hot wallet is empty, the cold wallet will refill the hot wallet.* It is extremely difficult to defend all attacks with current systems (Singhal and Ou 2017; Ding et al. 2018), especially for online systems like the hot wallet. Due to the existence of system vulnerability, exchanges need to set a threshold to determine how much funds should be reserved in hot wallets to balance theft risk and operational efficiency. The risk should be under control, even when the private keys of the hot wallet are leaked.

In this paper, for the first time, we provide a data-driven framework called SHORELINE to estimate the optimal threshold for a specific exchange, leveraging its historical data using machine learning techniques. There are two major components included as shown in the Figure 1: *a) Dynamic Networks Embedding.* In this component, we embed each cryptocurrency into a low-dimensional vector space temporally, based on the sampled sequences from temporal random walks on dynamic trading networks. *b) Optimal Threshold Estimation.* To estimate the threshold, we combine multiple data modalities from exchanges, such as the historical trading observations, withdraw and deposit history, currency embedding features. The goal is to predict the threshold of optimal reserves by a long-short-term memory network. We conduct extensive empirical studies to demonstrate the efficiency of the embedding method and the effectiveness of SHORELINE on real-world exchange data.

Related Work

Cryptocurrency and Exchange Security. A cryptocurrency transaction is a message propagated in the blockchain network signed by the private key of the sender, which specifies the volume of the type of currency transferred to a target address (Bamert et al. 2014). Once the message is signed by the private key, broadcasted to the blockchain, and acknowledged by other blockchain nodes, the transaction is persisted and considered settled, and anyone will not be able to tamper the records. This means when a private key is compromised or stolen, all the funds controlled by the key will be lost. Recently, most of the related works aim at tackling the problem based on encryption algorithms (Goldfeder et al. 2014; Gennaro, Goldfeder, and Narayanan 2016; Ziegeldorf et al. 2018). One recent approach aims at solving the problem from the statistical perspective. In (Jain, Felten, and Goldfeder 2018), the authors proposed a threshold control mechanism on the hot wallet to reduce the number of refilling from the cold wallet to the hot wallet, avoiding exposure expectation of cold wallet private keys during transfer. However, it neglects the operational differences among exchanges. The proposed SHORELINE provides a data-driven approach to enable exchange-specific thresholding, by considering historical trading and withdraw/deposit activities in an exchange.

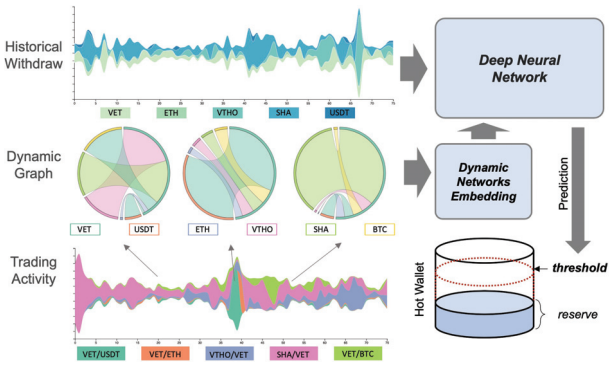


Figure 1: The overview of the proposed SHORELINE framework for optimal reserves threshold estimation.

Networks Embedding. There is a burst of embedding methods proposed to transform nodes (Perozzi, Al-Rfou, and Skiena 2014; Grover and Leskovec 2016), sub-graphs (Adhikari et al. 2017; Yanardag and Vishwanathan 2015; Adhikari et al. 2018), and graphs (Narayanan et al. 2017) features into low dimensional vector representations in the *static* condition. One of the popular pipelines is first sampling sequences based on random walks, and then embedding objects by skip-gram (Mikolov et al. 2013), which is a natural language processing model capturing the relation between one word and its co-occurrence context. (Wang, Cui, and Zhu 2016) proposed a deep auto-encoder based embedding method by adjacency matrix reconstruction. (Tang et al. 2015) used a specific embedding objective function combining first-order and second-order embedding similarity. Recent years have witnessed increasing attention to the general embedding of temporal networks (Li et al. 2017; Kumar, Zhang, and Leskovec 2018; Goyal, Chhetri, and Canedo 2018) such as social networks, citation networks, and economic networks. (Mahdavi, Khoshraftar, and An 2018) described a temporal extension of the random walk with skip-gram based embedding approaches. (Goyal et al. 2018) extends (Wang, Cui, and Zhu 2016) by introducing Net2WiderNet (Chang et al. 2015) tackling the appearance of new nodes. In (Li et al. 2018), a deep neural network with gated recurrent units is built to capture the changing trend of the adjacency matrix for the link prediction task. In this work, we propose an effective method to embed trading activity network, which is shown to greatly improve the threshold estimation performance.

Dynamic Networks Embedding

One cryptocurrency trading pair is two exchangeable digital currencies following a temporal rate. For example, BTC/VET pair represents the exchange relations between BITCOIN (Grinberg 2011) and the VECHAIN cryptocurrency. In one specific exchange, the amount of BTC is defined as trading volume v, and the amount of VET is defined as trading funds f. Formally, one trading pair at time t between currency i and j is defined as $p_{ij}^t = (v_i^t, f_j^t)$. A trad-

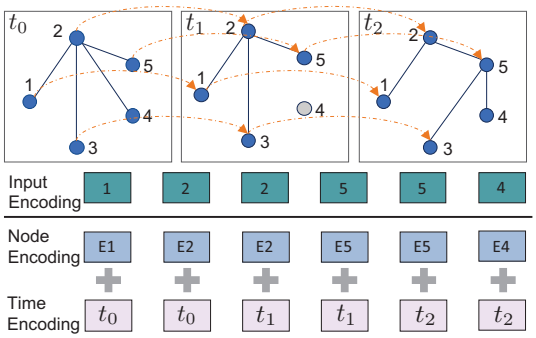


Figure 2: Inputs of neural networks for dynamic embedding.

ing network G_t at time t is constructed by all trading pairs with their associated currencies. The temporal network can be considered as the collection of snapshots at different time $G = \{G_{t_0}, G_{t_1}, \ldots, G_{t_n}\}$.

The temporal trading network keeps evolving as time goes on. The fluctuation of the market transitions will further affect users taking different actions, including deposit and withdrawal. In this section, we demonstrate our embedding approach for temporal networks to capture the evolution of digital currency transitions. We first sample a series of node (digital currency) sequences following the trading history from the temporal trading network constructed by the historical trading pairs and learn the contextual embedding of each digital currency based on deep neural networks.

Temporal Random Walk

It is a conventional approach to acquire the structural characteristics of a graph based on sampled nodes sequences by random walks (Sarkar and Moore 2011). Recently, the random-walk-based node embedding becomes prevalent and well-explored due to its low calculation complexity for static graphs (Perozzi, Al-Rfou, and Skiena 2014; Grover and Leskovec 2016). In order to extend the static embedding to dynamic embedding, it is necessary to design a temporal random walk strategy sampling nodes temporally to capture dynamic patterns.

We demonstrate our temporal walk procedure in Figure 2. The temporal random walk sequence is $1_{t_0} \to 2_{t_0} \to 2_{t_1} \to 5_{t_1} \to 5_{t_2} \to 4_{t_2}$. The temporal random walk requires a potential temporal transition from one node at a specific time step to nodes in the future. The temporal edges connect snapshots at different time steps together as an integrated graph. We create directed temporal edges from historical nodes to their future states between adjacent snapshots with weights derived based on adjacency considering chronology. Formally, we define the weight of one temporal edge as:

$$\text{weight}_{n_t \to n_{t+1}} = \gamma \sum\nolimits_{(n_t, j) \in G_t} \text{adj}(n_t, j),$$

where $\gamma \in (0, 1]$, $n_t \in G_t$, and $n_{t+1} \in G_{t+1}$. The weight of the edge from node n_t to n_{t+1} is proportional to the weight summation of edges with n_t as one endpoint, which means that there is an assigned probability to walk through n_t to its adjacent future state.

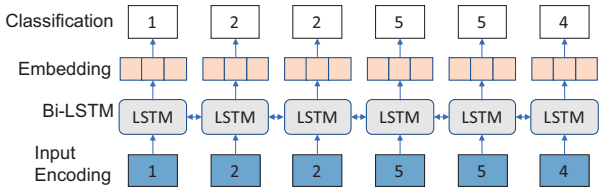


Figure 3: Illustration of contextual node embedding.

After temporal integration, we can adopt random walk strategies for static graphs (Grover and Leskovec 2016) as temporal random walks. Let n_i denote i_{th} node in the walk, starting at n_0. Then generation probability distribution of nodes n_i can be defined as:

$$\Pr(n_i = x \mid n_{i-1} = k,\ n_{i-2} = u) = \beta_{pq}(u, x)\frac{g_{kx}}{Z},$$

$$\beta_{p,q}(u, x) = \begin{cases} 1/p, & \text{if } d_{ux} = 0 \\ 1, & \text{if } d_{ux} = 1 \\ 1/q, & \text{if } d_{ux} = 2 \end{cases},$$

where g_{vx} is the weight of edge (k, x); Z is a normalizing constant; d_{ux} is the shortest distant between u and x; p and q are two parameters for the random walk preference.

The weight g_{kx} is assigned based on the trading information, which is proportional to trading volume v. It can be defined as:

$$g_{kx} = v_{k,x} / \sum\nolimits_{j \in \text{adj}(k)} v_{k,j},$$

where j is all currency can be exchanged with currency k.

The inputs of embedding are obtained by element wise summation of time encoding and node encoding indicating the temporal location of each sampled node as shown in Figure 2, where both encodings are trainable linear transformations.

Contextual Embedding

In previous node embedding methods (Grover and Leskovec 2016; Wang, Cui, and Zhu 2016), nodes are embedded based on the co-occurrence pairs from sampling inspired by WORD2VEC (Mikolov et al. 2013). The embedding from these approaches preserve first- and second-order proximity (Zhang et al. 2018), but ignore the local structure. It is acceptable for the original WORD2VEC to embed words without considering the local order, since the distance could vary between two related words with different expression preferences. For example, the meaning is the same of sentences "I have a cute cat," and "I have a cat, and it is cute." The order and distance are different of sense related word "cute" and "cat" in the two sentences. It is easy to find similar sentences with same meaning but different structures out of a given training dataset. The embedding without considering the local structures benefits from the flexible language structure. However, such freedom can not be achieved by random walks once the graph is given. Therefore, it is important to consider the local structure in

the node embedding, which is usually neglected by previous work. In this paper, we use Bidirectional LSTM (Bi-LSTM) embedding nodes fusing features from its neighbors based on temporal random walk sequences to capture the graph structure from both spatial and temporal dimensions inspired by its successful applications in natural language process (NLP) tasks recently (Peters et al. 2018; Devlin et al. 2018).

As in Figure 3, given a sequence of sampled currency nodes from temporal random walks $(v_0, v_1, \ldots, v_n)$, a forward LSTM models the probability of the sequence by the probability production of $v_i, 0 \leq i \leq n$, which is determined by the occurrence of its preceded nodes $(v_0, \ldots, v_{i-1})$,

$$\overrightarrow{\Pr}(v_0, \ldots, v_n) = \prod_{i=0}^{n} \Pr(v_i | v_0, \ldots, v_{i-1}).$$

Similarly, a backward LSTM captures the context features in the opposite direction,

$$\overleftarrow{\Pr}(v_0, \ldots, v_n) = \prod_{i=0}^{n} \Pr(v_i | v_{i+1}, \ldots, v_n).$$

For simplicity, we use $\overrightarrow{\Pr}(v_i)$ and $\overleftarrow{\Pr}(v_i)$ to indicate the modeling probability based on forward and backward contexts. Consequently, we design a deep neural network based on Bi-LSTM modeling the probability of one sequence by context from both sides of v_i jointly by maximizing the following objective:

$$\sum_{i=0}^{n} \log \overrightarrow{\Pr}(v_i; \theta_x, \overrightarrow{\theta_{\mathcal{F}}}, \theta_S) + \log \overleftarrow{\Pr}(v_i; \theta_x, \overleftarrow{\theta_{\mathcal{F}}}, \theta_S),$$

where θ_x represents the trainable parameters for time encoding and node encoding; θ_S represents the trainable parameters of one fully connected layer with the Softmax activation function to classify the learned embedding, the hidden states of Bi-LSTM units, into one-hot node ID; $\overrightarrow{\theta_{\mathcal{F}}}$ and $\overleftarrow{\theta_{\mathcal{F}}}$ are trainable parameters for two direction LSTM layers.

The hidden states at each time step of Bi-LSTM are extracted as the embedding of each node with high-order information from its surrounding context. One node might have different embedding features in different contexts, so we take the mean of all embeddings.

Reserves Threshold Estimation

Trading histories include wealthy market fluctuation patterns, which motivate users to adopt different operations for profit. Two more time series are taken into account in this component, the raw historical records from the hot wallet including withdraws and deposits, and trading histories, including trading volume and funds details. Consequently, we design a deep neural network to estimate optimal reserves threshold combining different time-series and embedding features. Each sequence is fed into an individual LSTM branch first and then fused by one fully connected layer to get the threshold estimation.

We now introduce the problem setting of this paper:

Reserves Threshold Estimation. For an online hot wallet of an exchange with potential unobserved secure defects, given the trading history of pairs $\{p_{ij}^{t_0}, \ldots, p_{ij}^{t_n}\}$, the withdraw and deposit history $\{w_i^{t_0}, \ldots, w_i^{t_n}\}$ and $\{d_i^{t_0}, \ldots, d_i^{t_n}\}$, $i, j \in C$, the objective is to predict the optimal threshold μ_i which is the maximum reserve of currency i in the hot wallet to balance the potential loss due to the compromise of hot wallet private keys and operational efficiency, where C is the set of all supported trading pairs of the crypto-exchanges. The unobserved secure defects exist in most cases, otherwise it is unnecessary to develop the offline cold wallet anymore.

Objective Function. The output of the estimation framework is the thresholds of the hot wallet. Ideally, the lowest $\hat{\mu}$ can be assigned as the net withdraw amount nw. However, as the threshold grows higher, the potential theft risk also becomes higher when malicious third parties compromise the hot wallet. In other words, when the actual net withdraw amount is low, we could set a relatively higher threshold to satisfy potential accident demands. On the contrary, the threshold should be lower than the actual demands when the net withdraw volume is too high, even it will lead to more time-consuming refilling operations transferring assets from cold wallet to hot wallet. We now define the loss function of threshold estimation at a specific time given one sample:

$$\mathcal{L} = \sum\nolimits_i \mathrm{ReLU}(nw_i - \hat{\mu}_i)^2 + \alpha \hat{\mu}_i^2,$$

where $nw_i = \mathrm{ReLU}(w_i - d_i)$, $\mathrm{ReLU}(x) = \max(0, x)$, w_i and d_i are withdraw and deposit amounts of currency i. The objective function combines both security and operational efficiency considerations. The first component means that the operation cost emerges if the net withdraw amount is higher than the estimated threshold $\hat{\mu}_i$. The second component represents the security concern that all currencies retain online are risky to be stolen. The coefficient α is set to balance the loss from both situations. If $\alpha = 0$, then the loss from vicious attacks would not be taken into account, which means that one can preserve infinity amount of currency on the hot wallet, and the cold wallet is no longer needed.

Evaluation of Networks Embedding

In order to illustrate the efficiency of our proposed architecture, we start from comparing the dynamic embedding architecture with other baselines by three general datasets, which are usually evaluated by other embedding methods:

IA-Radoslaw-Email (Michalski, Palus, and Kazienko 2011) is a network describing internal email communication among employees of a company. The nodes represent employees and edges represent individual emails between two users. There are 167 nodes and 82.9k temporal edges recorded in total, and the period is 271 days.

Fb-Forum (Opsahl 2011) is a Facebook-like forum network extracted from an online community recording the users' broadcasting activates to groups in the forum. The period is 164 days with 899 nodes and 33.7k edges.

Fb-Messages (Opsahl and Panzarasa 2009) records the temporal messages among members of a Facebook-like online community. The time span is 164 days with 1.9k nodes and 59.8k edges in 216 days.

The three datasets are collected from communication networks, that can also be thought as the more general cases of the trading graph. Each communication can be considered

Dataset	IA-Radoslaw-Email						Fb-Forum						Fb-Messages					
Snapshot	1	2	3	4	5	mean	1	2	3	4	5	mean	1	2	3	4	5	mean
Deepwalk	0.793	0.829	0.823	0.830	0.810	0.817	0.793	0.787	0.788	0.826	0.730	0.785	0.796	0.677	0.703	0.667	0.690	0.707
node2vec	0.811	0.844	0.856	0.860	0.828	0.840	0.805	0.803	0.805	0.830	0.756	0.800	0.822	0.735	0.721	0.671	0.729	0.736
dynnode2vec	0.848	0.829	0.843	0.832	0.806	0.832	0.792	0.738	0.730	0.737	0.724	0.744	0.736	0.664	0.663	0.589	0.664	0.663
SDNE	0.755	0.768	0.765	0.737	0.778	0.761	0.759	0.750	0.800	0.788	0.792	0.778	0.764	0.746	0.798	0.781	0.756	0.769
DynEmb	0.750	0.753	0.771	0.757	0.781	0.762	0.754	0.770	0.793	0.776	0.783	0.775	0.773	0.724	0.791	0.785	0.747	0.764
GRUEmb	0.827	0.821	0.844	0.809	0.793	0.819	0.839	0.817	0.836	0.803	0.804	0.820	0.752	0.684	0.770	0.686	0.774	0.733
LSTM	0.895	0.901	**0.906**	**0.914**	0.873	0.898	0.904	0.882	0.870	0.907	0.902	0.893	0.779	0.797	0.769	0.787	0.801	0.787
LSTM_TW	**0.901**	**0.909**	0.897	0.909	**0.908**	**0.905**	**0.916**	**0.911**	**0.905**	**0.921**	**0.926**	**0.916**	**0.833**	**0.812**	**0.816**	**0.843**	**0.837**	**0.828**

Table 1: AUC scores of link prediction task. The best performance is highlighted.

one transaction from one currency to the other one. Moreover, the communication frequency could change drastically like the trading network.

Baselines

Architectures based on the skip-gram (Mikolov et al. 2013):

- **Deepwalk.** (Perozzi, Al-Rfou, and Skiena 2014) It is a conventional embedding method for static graphs. The node representation is attained based on a skip-gram neural network trained with sequences from random walks.
- **node2vec.** (Grover and Leskovec 2016) It is the general extension of DeepWalk. Two parameters p and q are used to control the preference of random walks.
- **dynnode2vec.** (Mahdavi, Khoshraftar, and An 2018). It is the extension of node2Vec from static graphs to temporal graphs. Compared with node2vec, the model is focus more on the latest snapshot when graphs changes drastically.

Architectures based on the deep auto-encoder:

- **SDNE.** (Wang, Cui, and Zhu 2016) SDNE is a deep embedding method for static graph embedding considering the first-order and second-order similarity.
- **DynGEM.** (Goyal et al. 2018) DynGEM can be considered as a temporary extension of SDNE by introducing (Chang et al. 2015) to handle the growth of the graph.
- **GRUEmb.** (Li et al. 2018) The network is proposed with the supervised link prediction task by the gated recurrent units.

Here are our proposed architectures:

- **LSTM.** This is the simplified version of our proposed embedding method. One Bi-LSTM layer for nodes embedding is implemented without temporal random walks using the same random walk samples of node2vec.
- **LSTM_TW.** The contextual embedding architecture with one Bi-LSTM layer for nodes embedding based on the node sequences sampled temporal random walks.

In the following evaluation task, nodes are embedded into 128 dimension vectors. All hyper-parameters are tuned by grid-search.

Link Prediction

The link prediction task is used to evaluate if the embedding method can capture temporal patterns from historical observations. The architecture is first trained given a series of historical network snapshots $\{G_0, G_1, \dots G_t\}$. We further train a logistic regression model to detect if an edge $e \in G_{t'}$ exists using the embedding based on observations from previous n snapshots, where $t' > t$. The edges which exist in $G_{t'}$ are set with label 1, and the same amount of negative edges are sampled randomly. We hold out 75% of samples from $G_{t'}$ for training a logistic regression model and the rest for testing. The area under curve (AUC) is the evaluation metric. We have five snapshots for training, $n = 5$, and the rest 5 snapshots for testing. Only the best performance is reported with various edge representations based on different combination of node features (Grover and Leskovec 2016). The AUC scores of all testing snapshots are shown in Table 1. We have the following observations:

- The proposed contextual embedding architectures LSTM achieves better performance than baselines, which supports our claim that the local contextual order is critical during embedding.
- The LSTM_TW outperforms simple LSTM without temporal random walking, achieving the best performance. It proves that sampled sequences from the proposed temporal random walk contain wealthy temporal information.
- The performance of baselines based on the deep learning auto-encoder can not beat the simple structure Deepwalk, including SDNE, DynEmb, and GRUEmb. It probably because the embedding non-linearity is too high to be utilized by downstream models such as the logistic regression due to the complex encoding structure. The desired embedding features should be superficial enough to be exploited by other models. It also motivates us to design the LSTM_TW decoding the embedding feature at the last layer by a simple dense classifier.

Based on the link prediction task, we show that the proposed LSTM_TW can achieve the best performance on general social communication datasets compared with other baselines. Thus, we further apply it in embedding cryptocurrencies, which can be considered as a special social network.

Evaluation of Reserves Threshold Estimation

In this section, we first describe the data and the experimental settings and then show the effectiveness of our proposed architecture. At last, we finish two case study to analyze the performance qualitatively.

Our real trading history and the withdraw deposit records are obtained from a real anonymous crypto-exchange to avoid privacy leaking from 12/20/2018 to 5/11/2019. The

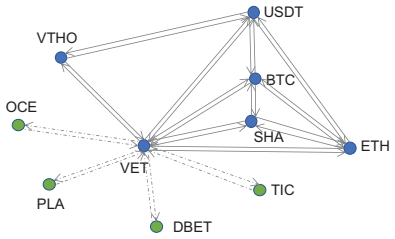


Figure 4: The trading graph constructed by trading pairs.

Model	Period	$\alpha = 10$	$\alpha = 5$	$\alpha = 3$	$\alpha = 0.1$	$\alpha = 0.01$
HA	1	0.517	0.275	0.177	0.037	0.032
	2	0.697	0.379	0.252	0.068	0.062
	3	0.838	0.455	0.302	0.080	0.073
	4	0.960	0.510	0.330	0.069	0.061
	mean	0.753	0.405	0.265	0.063	0.057
OTE-trade	1	0.739	0.417	0.209	0.035	0.016
	2	0.286	0.187	0.153	0.056	0.028
	3	0.212	0.193	0.168	0.062	0.027
	4	0.206	0.183	0.169	**0.046**	0.010
	mean	0.361	0.245	0.175	**0.050**	0.020
OTE-hist	1	1.085	0.792	0.479	0.036	0.013
	2	0.227	0.184	0.148	0.060	0.024
	3	**0.186**	0.181	0.174	0.061	0.019
	4	**0.180**	**0.175**	0.171	0.055	0.013
	mean	0.420	0.333	0.243	0.053	0.017
OTE-emb	1	**0.391**	0.229	0.164	0.036	0.017
	2	0.294	0.234	0.177	**0.053**	0.025
	3	0.315	0.273	0.222	0.064	0.027
	4	0.422	0.378	0.226	0.049	**0.008**
	mean	0.356	0.278	0.197	0.051	0.019
SHORELINE	1	0.395	**0.220**	**0.135**	**0.038**	**0.016**
	2	**0.142**	**0.138**	**0.129**	**0.053**	**0.016**
	3	0.194	**0.170**	**0.154**	**0.059**	**0.016**
	4	0.227	0.179	**0.156**	0.051	0.013
	mean	**0.240**	**0.177**	**0.143**	**0.050**	**0.015**

Table 2: The table shows the loss metrics $\mathcal{E}$ based on different α. The best results are highlighted.

available trading pairs are illustrated in Figure 4. We use historical trades of six days to predict the threshold of the further one day. The daily trading history is used to construct one snapshot of the temporal trading graph. We aggregate 6 hours of exchange records as one observation due to the sparsity problem. Also, we only use nodes 7 to 10 in the embedding step during temporal random walk because they are involved late and lack of observations. Several snapshots are shown in the Figure 1, we remove the real number to avoid commercial secrets disclosure. The threshold is estimated for maintaining the security and operational efficiency for the next day, because updating threshold frequently suggests high operation cost and high private key leakage risk of cold wallets. Thus, we use the mean of the net withdrawal in the next 24 hours as the reference value during training.

Performance Comparison

All records are divided into five periods chronologically. We use t_{th} period for training and testing on $(t{+}1)_{th}$ period with initialized weights from $(t-1)_{th}$ period if available. The embedding dimension is 2 because we only have 10 cryptocurrencies. The random walk length and number starting from each currency node are 10.

Evaluation Metrics. The first order objective function is used as the evaluation metrics,

$$\mathcal{E} = \frac{1}{n}\sum_{i=0}^{n}\sum_{j=0}^{c}\mathrm{ReLU}(nw_{ij} - \hat{\mu}_{ij}) + \alpha\hat{\mu}_{ij},$$

where n is the sample size, c is the currency number, $\alpha \in \{10, 5, 3, 0.1, 0.01\}$ denotes different levels of concerns for hot wallet security.

Here are the comparison models to fully evaluate each component of the framework.

Historical Mean (HA). The historical mean of net withdraws is used as the estimated threshold.

OTE-hist. One LSTM layer with 32 units is applied with only historical net withdraws as its input.

OTE-emb. Based on OTE-hist, the other LSTM layer with 32 units is applied to handle the temporal vectors of digital currencies' embedding.

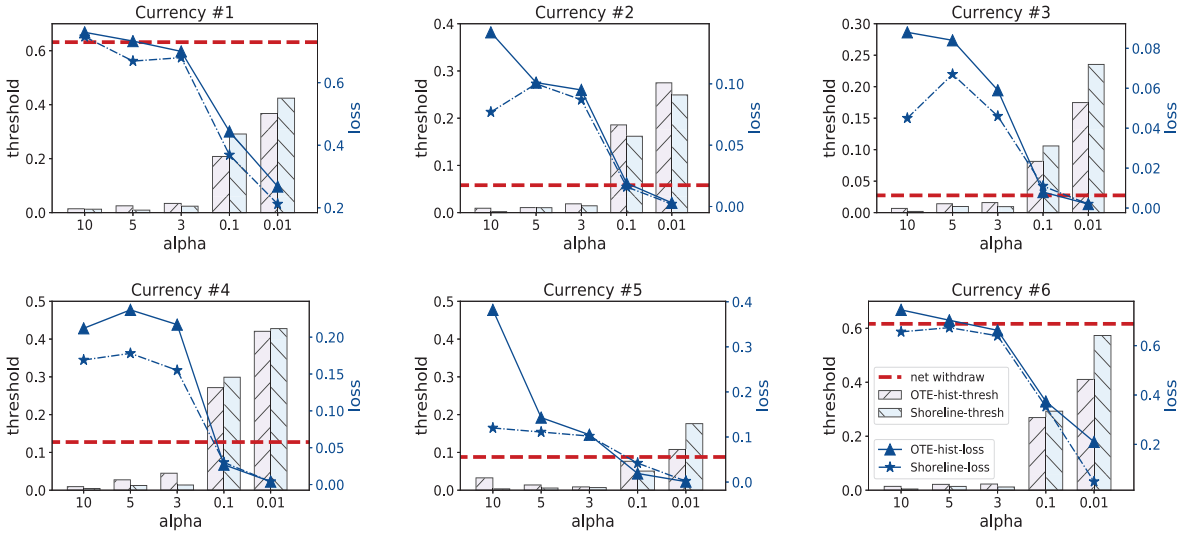
OTE-trade. It is the architecture with 3 LSTM layers with 64, 32 and 8 units to capture patterns in raw historical trading records of all trading pairs.

SHORELINE. The final optimal reserves threshold estimation framework combines all the above features and architectures by a dense layer. The dropout of 0.5 is used to avoid possible over-fitting problem since more features are considered as inputs.

The testing results are listed in Table 2. The SHORELINE can achieve the best testing performance under different security concerns α. The performance of OTE-trade is only slightly better than the historical mean, although the temporal trading networks for embedding are also derived from the trading history. It further shows the effectiveness of our proposed embedding methods. In general, as α becomes smaller, the loss $\mathcal{E}$ also becomes lower as Table 2.

Case Study: Effectiveness of α

Here we present several cases for the better understanding of our proposed framework in Figure 5. We compare the estimation results with one baseline OTE-hist, which only cooperate the net withdraw history to illustrate the effectiveness of the features from trading pairs. In general, the lower α is, the more the crypto-exchange values the operational efficiency by reserving more funds in online hot wallets to meet the withdraw demands. The online reserves decrease significantly when α becomes large meaning that security is the preliminary concern. In both situations, the SHORELINE framework can estimate the precise threshold, balancing both the security and operational efficiency of a crypto-exchange. The cases can be divided into two groups. 1) The threshold is higher than the average net withdrawal, when α is low, including currency $\#2$, $\#3$, $\#4$, and $\#5$. Note that the threshold is set for the next 24 hours, there is no guarantee that the demand can always be satisfied immediately given the mean net withdrawal amount. A higher amount of currency is usually desired to meet the accident significant withdraw if the system is secure. 2) The threshold is always lower than the mean of net withdrawal, such as currency $\#1$ and $\#6$. It is reasonable to estimate a low threshold when α is relatively large. As α grows, the withdrawal requirement



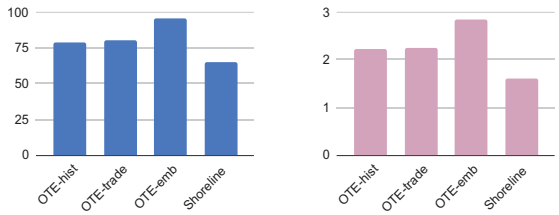
α(weight for security concerns) - $\mathcal{E}$(evaluation metrics) - T(threshold prediction)

Figure 5: The chart shows different cases of different currencies. The net withdrawal is the mean amount in 24 hours.

should not be satisfied by the pre-deposits in the hot wallet when the demand is too high. It is beneficial to minimize the loss if the system is corrupted.

Case Study: Theft Risks.

We further test the proposed SHORELINE given α by potential theft risks. We apply the threshold estimation framework on the currency $\#1$ based on the collected withdrawal and deposit history, which is the core currency of the exchange connecting with all the others. At first, the hot wallet is filled as the prediction threshold for high-frequency trading. When the reserve exceeds the threshold, the extra is transferred to the cold wallet. The cold wallet refills the hot wallet to the new estimated threshold, once the hot wallet is empty. In general, one exchange can afford one time of regular refill from the cold wallet to the hot wallet every day. Thus, we set $\alpha = 0.1$, where only a few unexpected refills will occur due to accident massive withdraws. The probability of unexpected refill is less than 5% per day for all models. We don't compare the performance of HA here because the probability is too high, which is around 50%. We compare the models based on two metrics, integration and surplus. The integration is the summation of currency hourly existing in the hot wallet in a day on average. The surplus is the average unnecessary currency remaining in the hot wallet after one day's trading. A higher integration value indicates that the currency exposes online for a longer time. And a higher surplus value represents more unused funds are reserved in the hot wallet. Therefore, the higher the two values are, the higher loss probably occurs if the system is compromised. Based on Figure 6, SHORELINE achieve the lowest integration and surplus values compared with other architectures, which means that the estimated threshold is efficient to balance theft risks and operational efficiency.



(a) integration (b) surplus

Figure 6: Risks comparison of different models, $\alpha = 0.1$.

Conclusion and Future Works

In this paper, we utilized trading history to estimate the reserves threshold of the online hot wallet for a crypto-exchange. We propose a new dynamic networks embedding method for temporal trading networks. Then we propose SHORELINE combining the embedding features and withdraw deposit history to estimate the final threshold. We conduct extensive experiments on the real dynamic networks and trading data to demonstrate the effectiveness of the proposed framework. In this paper, we only evaluate the proposed framework on the small- and middle-sized temporal graphs, since the size of trading networks are usually small. It could be an interesting extension if our embedding architecture could be applied to the large scale networks embedding such as social networks. Also, the embedding component and the threshold estimation component are trained separately. The performance could be improved if the proper combination of the two parts could be involved such as multi-modal learning algorithms. It is critical for finial scientists to make safe decisions if the meaning of the learned features is clear, and thus the feature interpretability is also a promising extension.

Acknowledgments

This material is based upon work supported by the VeResearch research gift from VeChain Foundation, National Science Foundation under Grant IIS-1749940, IIS-1615597, and Office of Naval Research N00014-17-1-2265.

References

Adhikari, B.; Zhang, Y.; Ramakrishnan, N.; and Prakash, B. A. 2017. Distributed representations of subgraphs. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, 111–117. IEEE.

Adhikari, B.; Zhang, Y.; Ramakrishnan, N.; and Prakash, B. A. 2018. Sub2vec: Feature learning for subgraphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 170–182. Springer.

Bamert, T.; Decker, C.; Wattenhofer, R.; and Welten, S. 2014. Bluewallet: The secure bitcoin wallet. In *International Workshop on Security and Trust Management*, 65–80. Springer.

Chang, S.; Han, W.; Tang, J.; Qi, G.-J.; Aggarwal, C. C.; and Huang, T. S. 2015. Heterogeneous network embedding via deep architectures. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 119–128. ACM.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ding, D.; Han, Q.-L.; Xiang, Y.; Ge, X.; and Zhang, X.-M. 2018. A survey on security control and attack detection for industrial cyber-physical systems. *Neurocomputing* 275:1674–1683.

Facebook. 2019.

Gennaro, R.; Goldfeder, S.; and Narayanan, A. 2016. Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security. In *International Conference on Applied Cryptography and Network Security*, 156–174. Springer.

Goldfeder, S.; Bonneau, J.; Kroll, J.; and Felten, E. 2014. Securing bitcoin wallets via threshold signatures.

Goyal, P.; Kamra, N.; He, X.; and Liu, Y. 2018. Dyngem: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273*.

Goyal, P.; Chhetri, S. R.; and Canedo, A. 2018. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *arXiv preprint arXiv:1809.02657*.

Grinberg, R. 2011. Bitcoin: An innovative alternative digital currency. *Hastings Science & Technology Law Journal* 4:160.

Grover, A., and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 855–864. ACM.

Jain, S.; Felten, E.; and Goldfeder, S. 2018. Determining an optimal threshold on the online reserves of a bitcoin exchange. *Journal of Cybersecurity* 4(1):tyy003.

Kumar, S.; Zhang, X.; and Leskovec, J. 2018. Learning dynamic embeddings from temporal interactions. *arXiv preprint arXiv:1812.02289*.

Larcheveque, E. 2018. *2018: A Record-Breaking Year for Crypto Exchange Hacks*. https://www.coindesk.com/2018-a-record-breaking-year-for-crypto-exchange-hacks [Accessed: Whenever].

Li, J.; Dani, H.; Hu, X.; Tang, J.; Chang, Y.; and Liu, H. 2017. Attributed network embedding for learning in a dynamic environment. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 387–396. ACM.

Li, T.; Zhang, J.; Philip, S. Y.; Zhang, Y.; and Yan, Y. 2018. Deep dynamic network embedding for link prediction. *IEEE Access* 6:29219–29230.

Mahdavi, S.; Khoshraftar, S.; and An, A. 2018. dynnode2vec: Scalable dynamic network embedding. In *2018 IEEE International Conference on Big Data (Big Data)*, 3762–3765. IEEE.

Michalski, R.; Palus, S.; and Kazienko, P. 2011. Matching organizational structure and social network extracted from email communication. In *Lecture Notes in Business Information Processing*, volume 87, 197–206. Springer Berlin Heidelberg.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Nakamoto, S., et al. 2008. Bitcoin: A peer-to-peer electronic cash system.

Narayanan, A.; Chandramohan, M.; Venkatesan, R.; Chen, L.; Liu, Y.; and Jaiswal, S. 2017. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*.

Opsahl, T., and Panzarasa, P. 2009. Clustering in weighted networks. *Social networks* 31(2):155–163.

Opsahl, T. 2011. Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. *Social Networks*.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710. ACM.

Peters, M.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2227–2237.

Sarkar, P., and Moore, A. W. 2011. Random walks in social networks and their applications: a survey. In *Social Network Data Analytics*. Springer. 43–77.

Singhal, A., and Ou, X. 2017. Security risk analysis of enterprise networks using probabilistic attack graphs. In *Network Security Metrics*. Springer. 53–73.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, 1067–1077. International World Wide Web Conferences Steering Committee.

Wang, D.; Cui, P.; and Zhu, W. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 1225–1234. ACM.

Yanardag, P., and Vishwanathan, S. 2015. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1365–1374. ACM.

Zhang, D.; Yin, J.; Zhu, X.; and Zhang, C. 2018. Network representation learning: A survey. *IEEE transactions on Big Data*.

Ziegeldorf, J. H.; Matzutt, R.; Henze, M.; Grossmann, F.; and Wehrle, K. 2018. Secure and anonymous decentralized bitcoin mixing. *Future Generation Computer Systems* 80:448–466.