SWIoTA: Anomaly Detection for Distributed Ledger Technology-Based Internet of Things (IOTA) Using Sliding Window (SW) Technique

Sathish Kumar, Norman Ahmed, Anastasios Bikos

SWIoTA: Anomaly Detection for Distributed Ledger Technology-based Internet of Things (IOTA) using Sliding Window (SW) Technique

Sathish A.P. Kumar[1] , Norman Ahmed[2] and Anastasios Bikos[3]

[1] Cleveland State University, Cleveland, OH, USA (s.kumar13@csuohio.edu)
[2] Air Force Research Laboratory, Rome, NY, USA (norman.ahmed@us.af.mil)
[3] University of Patras, Patra 265 04, Achia, Greece (mpikos@ceid.upatras.gr)

Abstract. IOTA is a Digital Ledger Technology (DLT) prototype for IoT applications that has attracted a rising popularity in recent years. One issue that acts as obstacle to its widespread adoption are the cybersecurity concerns. Some of the security concerns in IOTA include Denial of Service (DoS) double spending, parasite attacks, and DDoS attacks. In this work, we developed a Machine-Learning (ML) approach to create security threat index that can be utilized to proactively provide defenses to the IOTA decentralized infrastructure as well as individual nodes against potential compromises. Our approach is established on the sliding window customized technique to classify the data generated from the DAG-based nodes for cybersecurity anomaly detection. To validate the approach, we implemented "DoS attacks" threat model in the DLT-based IoT environment using Raspberry Pi devices and experimented our security methods and algorithms in this environment. The preliminary experimental results are promising.

Keywords: Distributed Ledger Technology, Internet of Things, Tangle, Cybersecurity, Sliding Window Technique, Anomaly Detection, Machine Learning.

1 Introduction and Background

The concept of Blockchain technology has got traction recently due to the increasing utilization of decentralized systems and the spreading need for integrity in the data management [3, 20]. The Blockchain [6] technology is based on an innovative data structure that is feasible to be used as a ledger for many applications. Bitcoin and Ethereum [24] are the most popular implementation of the cryptocurrency concept, which run on distributed ledgers. While DLT was originally used for recording financial transactions through bitcoins as well as other cryptocurrencies [6], this technology is being used in several domains such as Internet of Things (IoT) [7, 9-12, 22]. One such recent and versatile research effort that uses the distributed ledger protocol is Tangle [16]. Tangle is utilized as cryptography on the Internet of Things (IoT) to store P2P transactions. As shown in Figure 1, Tangle is a Directed Acyclic Graph (DAG) where a vertex, defining a transaction, has two ancestors, serving as the transactions it acknowledges. As per its protocol, a Proof of Work (PoW), or a Proof of Stake (PoS – an alternate solvability of the consensus based on intrinsic properties, such as the number of obtained tokens), has to be fulfilled when adding a transaction to the Tangle. This should hinder an adversary from doing network spamming. However, it is not yet clear the amount of cybersecurity impact from the PoW/PoS instantiation in the Tangle. IoT connects various physical devices that we use on a

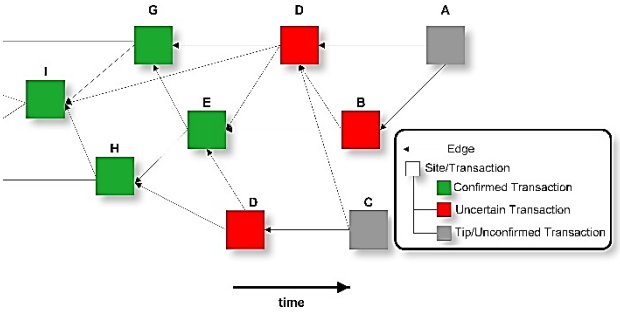

Fig. 1. An example instance of a Tangle

daily basis and enable them to interact with each other through the Internet. This ensures the intelligence of the devices [4, 5, 8]. IoT is used in a variety of domains such as military, healthcare, logistics, utilities and smart cities [1]. IoT is expected to revolutionize the future innovations especially related to Industry 5.0 . Use of IoT is expected to rise on an exponential basis over the coming years. Security is an important issue due to a huge number of devices that are linked to the Internet and the massive attack surface area associated with it [2, 21]. Many of these physical devices from an Internet of Things perspective are easy targets for the intrusion due to the huge attack surface area and moreover they rely on exterior resources and are often left disregarded. Due to the integration of DLT with IoT, cybersecurity and trust management in the consensus scenarios is a very important consideration [13-15, 17-20, 23, 25-26, 30].

Following are the contributions of this work: a. Prototype a formal sliding window-based approach (SWIoTA) to construct a security threat index for each DAG-based device to achieve anomaly detection, in real time, and b. evaluate the performance of the previous anomaly detection approach using detection with outlier factor and Mahanabolis distance metric.

The rest of the paper is organized as follows. In section 2 we illustrate the security problem and its importance. Methodology is described in Section 3. Section 4 describes the experimentation and performance evaluation results. Future work recommendations are described in Section 5. Finally, Section 6 describes the conclusion.

2. Problem Statement and Rationale

2.1 Problem Statement

While there are several attacks possible in the DLT-based IOT, per literature review one of the important attacks that needs attention is Denial of Service (DoS) attacks. In this work, leveraging our earlier work [4, 31-36], our objective is to implement anomaly detection algorithms to detect attack and to protect the DLT framework-based IoT network from DDoS attacks.

2.2 Rationale for DoS Threat Model

The proposed approach leverages our private Tangle network for IoT as a test platform and integrate the proposed security methods into the existing DLT-based framework for IoT.

While setting up the Private Tangle network, the Hornet nodes and participating clients are assumed to be added using a permission-based approach. As the name suggests, in the private tangle setup, the participating nodes does not connect to public Internet directly. Only participants that are in the local area network are allowed to participate. The IP addresses of the hornet nodes belong to a private network domain. By configuring port forwarding on the gateway, it is possible to allow geographically remote participants, but still each remote participant needs to know the IP/port of the gateway, and it is still permission based. In such a Private Tangle setup, due to the permission-based nature, the administrators could relax the security precautions against the local area network participants. For instance, they might allow any local IP to request remote-proof-of-work. Our setup followed the Private Tangle based network and assumed that local participating/transacting clients are trusted implicitly. We assume that one of the local devices is compromised and started to run an attacker script. The attacker that compromised the local device is utilizing PyOTA client library. With PyOTA python library [29], attacker can send Zero-value-messages to any Hornet node in the Private tangle. Our attack uses Zero-value-messages, but in a threaded manner. From the attack script, we create as many threads as possible and each thread sends a zero-value-message request to the same hornet node. Without threaded approach, PyOTA client would wait for a response before sending the next request. But with threaded request, attacker can send hundreds of requests in a couple of seconds and keep the target Hornet nodes resources tied for significant amount of time.

2.3 The SWIoTA blockchain security contribution

Utilizing the previous underlying assumptions, we formally depict SWIoTA, a dynamic, lightweight, and portable cybersecurity framework which exploits the blockchain concept to perform purely on-line (dynamic & stateless) and distributed anomaly detection on resource constraint devices such as Internet-of-Things (IoT).

Through a centralized coordinator entity (Compass), inside our IoTA network topology, our framework model is being trained and provides detection decisions for each connected node based on their own locally examined resource behaviors. The introduction of the Sliding Window (SW) emerges as a vital fully dynamic component which holds the properties of a moving window with short memory across the time axis of the evolving node resources. Co-deploying the SW with highly accurate Machine Learning operations that are based on output-code classifiers can characterize anomaly behaviors, other than technical losses, with much higher sensitivity. Our unique contribution is the adoption of the SW technique alongside customized classifiers. This deployment seems prominent and energy efficient for lightweight IoT.

3. Methodology

To invoke the multiclass strategy to generate our labelled security threat index, we follow the notation of the design of continuous codes using Quadratic Programming, which is primarily an optimization problem. Output-labelling based methods often consist of representing each class with specific index. During detection, the indexes are arrived based on the learning from the input data. During the prediction time, the classifiers used during the detection time can be used to project new points and the class that is nearest to the points projected will be used for the prediction.

3.1 Error-Correcting Output-Code multiclass strategy

For convenience we use the square of the norm of the matrix (instead the norm itself). Our ML classifier H(x) is constructed from a code matrix M and a set of binary classifiers $\bar{h}(x)$. The matrix M is of size k × l over R where each row of M is corresponding to a class y ∈ Y. Provided an instance x, the classifier H(x) predicts the label y which maximizes the confidence function K(h(x), Mr), H(x) = argmaxr ∈ y{K(h(x), Mr)} . Since the code is over the real numbers, we can assume here without loss of generality that exactly one class concentrates the maximum value according to the function K. To simplify the equations, we denote by τi = 1yi − ηi, the difference between the correct point distribution and the distribution obtained by the optimization problem. Thus, the general dual optimization problem can become therefore:

$$max_{\tau}\ \partial(\tau) = -\frac{1}{2}\beta^{-1}\sum_{i,j} K\left(\bar{h}(x_i), \bar{h}(x_j)\right)\left(\bar{\tau}_i \cdot \bar{\tau}_j\right) - \sum_i \left(\bar{\tau}_i \cdot \bar{\beta}_i\right) \quad \text{.... (1)}$$

subject to:

$$\forall i\ \bar{\tau}_i \leq \bar{1}_{yi}$$

and:

$$\bar{\tau}_i \cdot \bar{1} = 0$$

and our (optimal) classification rule becomes:

$$H(x) = argmax_r\left\{\sum_i \tau_{i,r}\, K\left(\bar{h}(x), \bar{h}(x_i)\right)\right\} \quad \text{.................................. (2)}$$

The general equations for designing output codes using the optimization problem described above, also provides, as a special case, our algorithm for building multiclass Support Vector Machines, in order to label the security threat index.

3.2 Classification Problem

Our anomaly detection scheme follows the practice of Output-code multiclass strategy (also referred as "error-correcting output codes") to output the label indexes

based on the closeness of those labels. This class strategy allows to approximate a multi-class classification problem with a binary classifier. Each class is converted into a code of "0" s and "1" s. The length of the code is known as the code size. The codes associated with the classes are stored in the codebook. When a new sample arrives, the label's code is extracted from the codebook. Then, each classifier is trained on the corresponding part of the code, which can be either a 0 or a 1. For the prediction part, the classifier outputs a probability. These probabilities are compared to each code in the codebook, and the label that is the closest is selected as the most likely class. Manhattan distance is being utilized to determine that closeness.

Inside our paradigm use case we performed Multiclass Classification with Error-Correcting Output Code-based classifier. For instance, since we need to obtain 10 unique labels, or security threat indexes (from 0-10), we will have a target cardinality length space of 10 classes.

3.3 The Sliding Window (SW) Concept

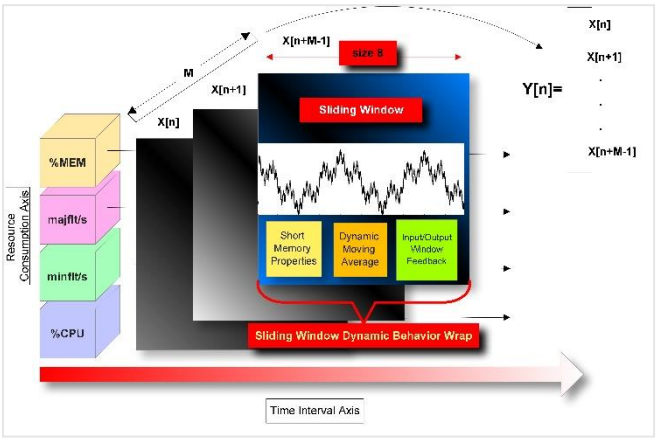

Fig. 2. The Sliding Window functionality.

Our fully ad-hoc Sliding Window technique much resembles the *moving average* calculation process. In statistical analysis, a moving or rolling average is an estimation to analyze data points by interpreting a series of averages, or means, of versatile subsets of the complete data set. Our approach holds additional dynamic properties such as *sliding average* property, *memory property* and *left/right feedback* (at the inputs/outputs of the window). As we can illustrate at Figure 2, the "moving" window stochastically slides across the time axis (with an empirical static size of 8

value), inputs the resource monitoring metrics and fully-feeds back in recursive cycles all its convoluted results. Notably, although the size of the window is 8, it moves sequentially (with step size of 1 value) among the previous axes. Such behavior of our SW can easily approximate a type of dynamic convolution and it can be viewed as an instance of a low-pass filter utilized in signal processing.

```
PROCESS STEPS:
1 INPUT ARRAY with size: {arbitrary length x 4}
2 DEFINE sliding window {W} with size=8
3 FOR EACH ARRAY ROW (with STEP=1):
4       APPLY sliding window {W}
5       {W} OUTPUTS correlation coefficients of
        ARRAY values
6       GET upper triangular part of step 5 output
7 1D interpolate ALL VALUES from step 6
  from +0 to +10
8 INPUT data AND ARRAY samples to ML
  Classifier
9 OUTPUT Security Threat Index (label) PER
  ARRAY   ROW
```

Listing. 1. Threat Index Computation Pseudocode

3.4 Threat Index Computation Process

We input resource consumption metrics array for each DAG-based node of the ledger. We output the same array but labelled, this time, with a security threat index (as indicated in Listing 1), which indicates the security severity of a presence or not of any security threat. It can be utilized for anomaly detection and security threat attack prevention on the IOTA. We retrieve the correlation coefficients between neighboring values in the ARRAY and get the upper triangular part, using the sliding window custom technique. The benefit of this sliding window concept is that (1) it converges stochastically over the Monte Carlo distribution probability density function of the time axis expansion of the DAG, and (2) it covers inter correlation features extraction from the array data samples that are neighboring enough, thus have same anomaly or idle behavior. Our correlation coefficients formula

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 (\sum y)^2]}} \quad \ldots\ldots. (3)$$

In equation 3, r is correlation coefficient, x represents values of the x-variable in a sample, y represents values of the y-variable in a sample and n is the number of samples.

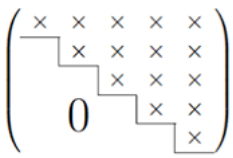

Fig. 3. Correlation Matrix

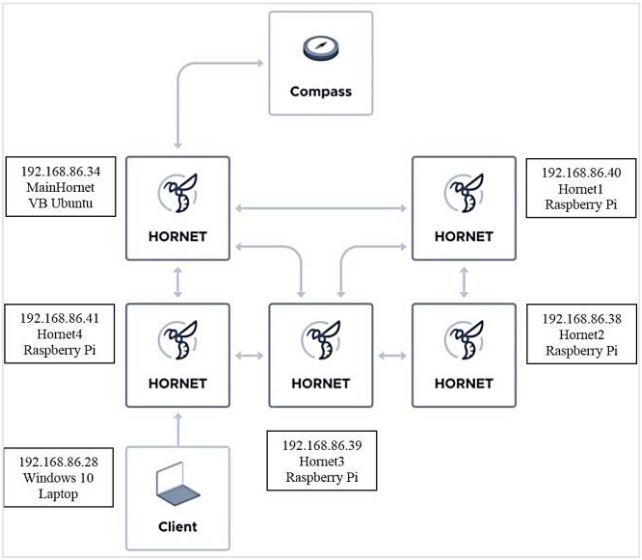

Fig. 4. Experimentation Setup

pre-training data. The aim is to create pre-training indexed features. We collect all correlated values onto a new matrix, as shown in Figure 3. We then 1D interpolate these values into our numerical index scale from +0 to +10. Finally, we input the interpolated data (such as pre-training features) and training array samples to the Multiclass Classifier (H(x)) and multilabel algorithm ML library. The goal is to perform a multiclass classification task with more than two classes and label the Threat Index per each array row.

4 Experimentation and Results

In this particular section, we demonstrate our experimental setup, threat model implementation, analysis, data collection and experimental results.

4.1 Private Tangle Setup

It is an IOTA network that we have full control over compared to public network. This Tangle will only connect to the nodes that we provide. Our private network is set up using the open source technology that make up the public IOTA networks. Compass as shown in Figure 4, is the main coordinator, that we implemented using the open source software. For our experiments needs we used four Raspberry Pis and one virtual machine Ubuntu device. We initiated the experiments by installing the hornet that will include the Compass. As shown in Figure 4, the compass was deployed on the virtual box with Ubuntu. Then we installed the Hornet for the Raspberry Pi after reassuring that the Hornet that is already installed is not being executed. Figure 4 depicts our main network topology to instantiate our IOTA experiments using the Raspberry Pi's.

4.2 Denial of Service Threat Model Implementation

	%CPU	minflt/s	majflt/s	%MEM	Threat Index
0	1.49	0	0.5	4.33	2
1	1	0	0	4.33	1
2	2	0.5	1	4.33	2
3	1.5	0	0	4.33	2
4	1	0	0	4.33	1
5	2.5	0	0	4.33	3
6	1.49	0	0	4.33	2
7	2	0	1	4.33	2
8	1	0	0	4.33	1
9	1.5	0	0	4.33	2
...	...	...	...	...	...

Fig. 5. Input Parameters

Time interval	%CPU	minflt/s	majflt/s	%MEM
0	1.49	0	0.5	4.33
1	1	0	0	4.33
2	2	0.5	1	4.33
3	1.5	0	0	4.33
4	1	0	0	4.33
5	2.5	0	0	4.33
6	1.49	0	0	4.33
7	2	0	1	4.33
8	1	0	0	4.33
9	1.5	0	0	4.33
10	1	0	0	4.33
11	0.5	0	0	4.33
...	...	...	...	...
288	1	0	0	4.45
289	1.5	0	0	4.45
290	1	0	0	4.45
291	0.5	0	0	4.45
292	2.5	0	1.5	4.45
293	1	0	0	4.45
294	6.5	0	0	4.45
295	1	0	0	4.45
296	1.5	0	0	4.45
297	2.5	0	1.5	4.45
298	1	0	0	4.45

Fig. 6. Output Threat Index

To emulate DoS attack, we selected one Hornet on the network and then launched as many transactions as we can in a short period of time. First, we tried to send transactions, then we modified our code to send multiple transactions in a short period of time. The modified code below will send as many transactions as it can in a given time. It creates multiple threads to send transactions so that it does not have to wait for the reply of each transaction before sending the next one. We will also be collecting time for each transaction, along with it we will also keep track of how many transactions we send out. The duration of the loop can be changed in the code and IP address target could be changed using the code as well. In one second we were able to send 18 transactions. During this setup, we were targeting the hornet on the Virtual Box, which at the time of setup was using 4 CPU cores. This is why the return time on each transaction is exceptionally low. When setting up an actual attack, it is better to target a hornet on one of the Raspberry PIs because it is a slower device. When configured for 14 min, Raspberry Pi will usually take more than 2mins to respond to each transaction.

4.3 Data Collection

We used pidstat command to collect the data. We also captured the heartbeat data for the hornet by using journalctl command. We then used nethogs to collect network traffic data. All these commands wrote to a log file or a text file. Hence, we created a script to extract the data to a more usable format. As shown in Figure 5, the data that we captured include CPU usage, memory usage, minor faults, and major faults. These explicit parameters were used to generate threat Index, which is explained in the following paragraphs. In terms of aggregating ground-truth labels, or correctly mapping identified security label(s) between benign and non-benign states, it is the mathematical formality of the SW algorithm itself (see Section IV.D) that is able to generate such outputs.

4.4 Anomaly Detection Using Sliding Window Technique

We executed the PIDSTAT command from the IOTA customized environment to retrieve real time /proc/stat resource monitoring status for our scenario's Hornet

components. The generated results (filtered) are tabulated below for 300 time slot intervals as shown in Figure 5. These time intervals typically last for 3 to 4 Seconds. The first column represents the timeline interval for the experimentation. In IOTA and DLT-based ledger framework(s) the time evolution of the Tangle and the Graph is both of paramount importance and a generic feature of the IOTA itself, because as time 't' progresses, even more transactions are logged (confirmed state), whereas others become unconfirmed or invalidated. Inside the scope of our mathematical concept setup we mainly focus, or take into consideration four resource metrics (%CPU, minflt/s, majflt/s and %MEM).

We will try to solve a multi-variate classification problem to generate a single label index, which we name as Threat Index, based on Machine Learning approach. Following the above customized mathematical technique and the Python ML libraries, we obtain the Threat Index per ARRAY ROW. We follow the below steps and incorporate a customized mathematical strategy to label the ARRAY rows with the security threat index (with numerical range from +0 to +10); where *0 to 3 is Low Security Risk, 4 to 6 is Medium Security Risk, and 7 to 10 is High Security Risk*. Following the above customized mathematical technique plus the Python ML libraries we obtain the security threat index as shown in Figure 6.

Figures 7 and 8 illustrate our experimental results for a custom IOTA deployment scenario using four Hornet devices and the Compass. The horizontal axis corresponds to the time interval during the course of the experiments. The vertical axis represents the newly generated threat index. Figure 7 depicts the idle, or no attack presence state. Specifically, this figure projects what is a typical running experimental state of a Hornet IOTA device node, without any security attack occurrence scenario. We could map this phase as a before and/or after an attack would take place. During the time interval from 920 (Second(s)) till 1100 (Second(s)), we subjected the Tangle network with DoS attack. After DoS attack is carried out, as shown in the Figure 8, it is clearly noticeable that the security threat index appears to climb at maximum threshold of 10, which flags as an anomaly. As we will evaluate our detection scheme in the next subsection, we concentrate on the accuracy sensitivity metrics to derive the ratio of false positives as well as f-measurement to estimate the correct mapping of security anomaly indexes among correctly identified security anomaly traits.

SWIoTA framework successfully identifies the attack duration segment within its time frame. The rate of correct mappings between technical losses and non-technical losses (anomalies) also seems quite promising, as it is observable in same figure that some early spikes before the attack takes place, or even after the attack correspond to technical incident(s) and not necessarily security incident, thus not being able to raise an alarm.

4.5 Performance Validation Experiments

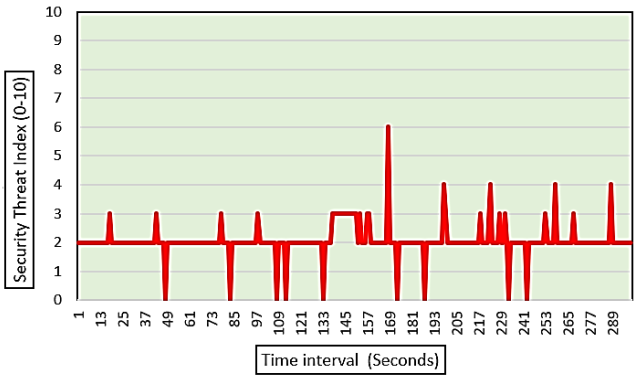

Fig. 7. Idle State

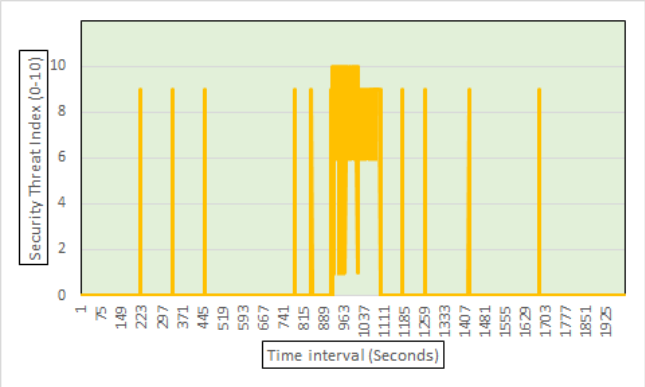

Fig. 8. Attack State

From an evaluation perspective, it is important for the applications to check if a new data point is in the same distribution as existing data point, or if that is an outlier (anomaly). This is especially important if the application is related to anomaly detection to detect abnormal or unusual observations. From an anomaly detection perspective, the outlier data points cannot be part of dense cluster and the classifiers/estimators assumes them to be part of the low- density regions. The objective of the outlier detection is to delineate core of regular observations from the outliers. It is worth noticing that these ML anomaly prediction techniques encompass PCA analysis and are considered the most innovative techniques currently deployed for network traffic threat detection/isolation. The processing of data with huge set of variables is particularly challenging. While there are many methods for dimension reduction, the most popular approach that is being applied is principal component analysis (PCA). We compare the performance analysis of our “sliding window” and ML technique with PCA/LOFs “conventional” approaches [28]. Hereby, we

input/utilize to these libraries our same input monitoring data metrics file from the 4 Hornets. As shown in Figure 8, the regular observations and abnormal observations cluster separately as expected. This validates the threat index detected using our sliding window approach. Namely, the observations inside the green dashed 2D surface belong to the idle (non-Anomaly) threat indexes, or states, whereas *anything outside is considered an Anomaly observation.*

Lastly, we collate & correlate the observation dynamics between our sliding window (SW) technique across the Mob distance in terms of their Hamming Distance (Positive Predictive Value, or PPV).

The Mahalanobis distance metric: One of the popular technique used for evaluating the cluster and classifier analysis is The Mahalanobis distance metric. We implemented this metric to measure of our classifier that distinguishes “normal” vs “anamoly” classes.

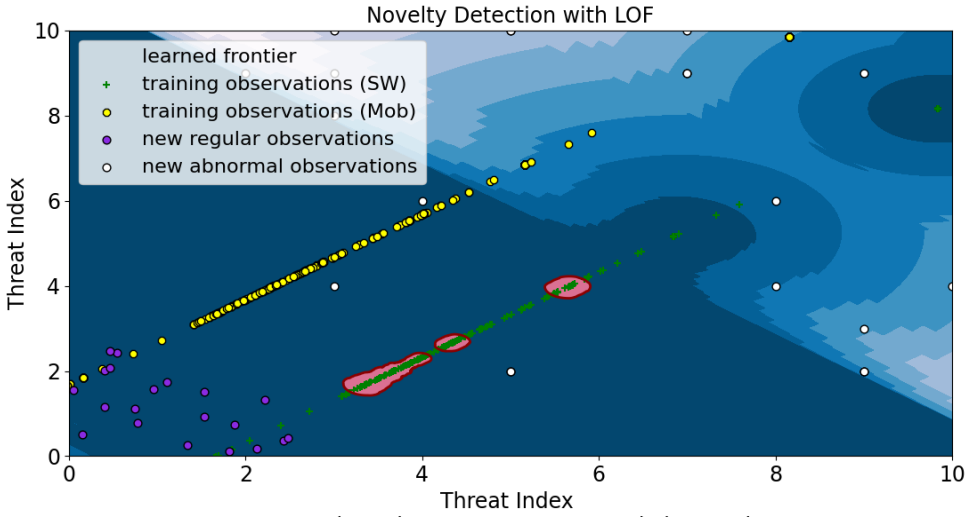

Fig. 9. Novelty detection with Local Outlier Factor (featuring Sliding Window technique vs Mahalanobis distance)

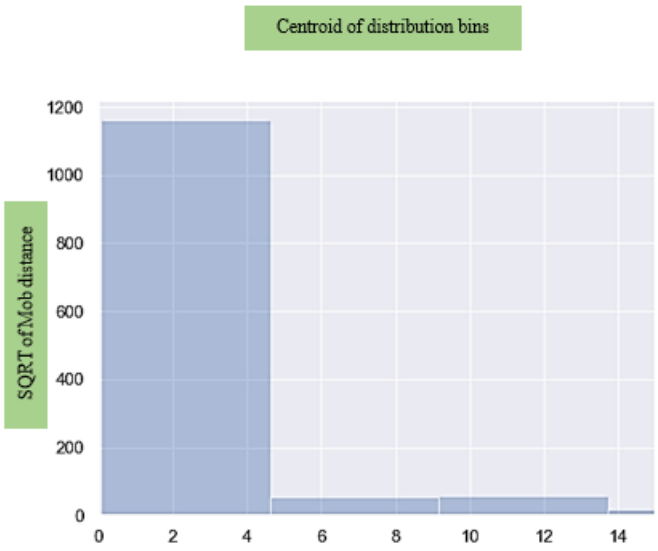

Fig. 10. Square of the Mahalanobis distance

2) Threshold value to detect an anomaly: The square of the Mahalanobis distance to the centroid of the distribution should follow a $\chi 2$ distribution, if the assumption of normally distributed input variables are fulfilled. This is also the basis of assumption behind the above calculation of the "threshold value" for flagging an anomaly. Since we cannot assume the input variables to be normally distributed, it is better to visualize the distribution of the Mahalanobis distance comparing it to threshold values in order to flag anomalies. As shown in Figure 9, we visualized the square of the Mahalanobis distance, to ensure that it follows a $\chi 2$ distribution. Then we visualized the Mahalanobis distance itself as shown in Figure 10. Based on the distributions shown in Figure 11, the computed threshold value comes to 3.8, which is 3 standard deviations from the center of the distribution and is used to flag an anomaly. We can then save the Mahalanobis distance, as well as the threshold value and "anomaly flag" variable for both training and testing data in a data frame. That way any observed distance above the threshold value can be flagged as an anomaly. As shown in Figure 12, we plot the computed anomaly metric (Mob distance) and confirm when it crosses the anomaly threshold. Based on our proposed approach and the comparison models, it seems illustratively enough we have strong accuracy, threat detection estimation, and predictability as we fall inside the same attack time margin on both figures, Figure 8 and Figure 12 during the time index interval between 912 and 1100, when the attack takes place against the IOTA hornets and as shown in Figure 12, the Mob distance lies above the estimated red line threshold (event indicated as Anomaly), and as shown in Figure 8, our estimated threat index is again high enough to be classified as a real attack (around 10). Finally, Figure 13 confirms the latter. Inside the former figure, SW stands for Sliding Window, Mob dist is the practical implementation of Novelty Detection/Local Outlier Factor(s) (using Mahalanobis distance), and Thresh

is simply the objective Anomaly indicator for both processes. Any indication above that limit is strictly considered as an Anomaly. The reason we are co-deploying SW vs Mob technique is that we need to exploit a comparison scheme for our methodology, as we will illustrate next.

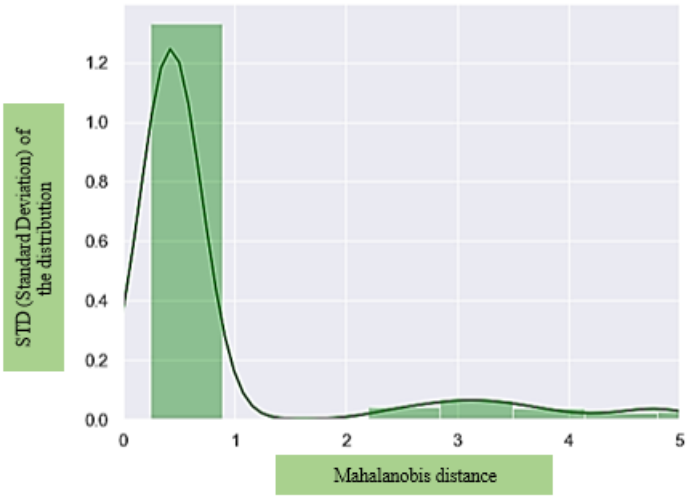

Fig. 11. Mahalanobis distance

From visually observing the two (sub)plots in Figure 12, the level of coincidence (Hamming distance of Threat Indexes) between two fundamentally versatile procedures is remarkable. Thus, we can derive the accuracy of the sliding window technique seems promising. In fact, how accurate an intrusion detection system, alike our sliding window methodology for cybersecuring the IOTA ledger, could be evaluated? Retrieved from the research literature [27], several accuracy metrics such as False Positive Ratio (FPR), True Negative Ratio (TNR), Detection Rate (DR), F1 score and Misclassification rate can be introduced.

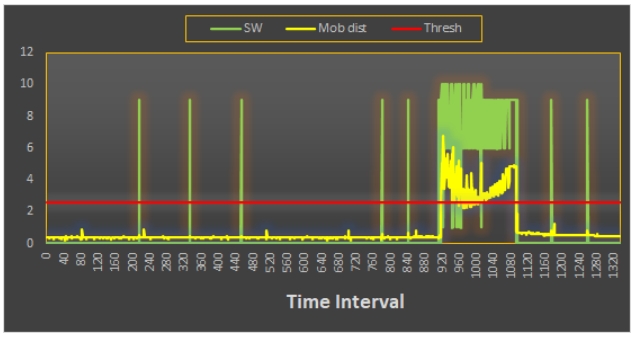

Fig. 12. Comparison Plot of our Sliding Window technique vs Mahalanobis distance vs Threshold

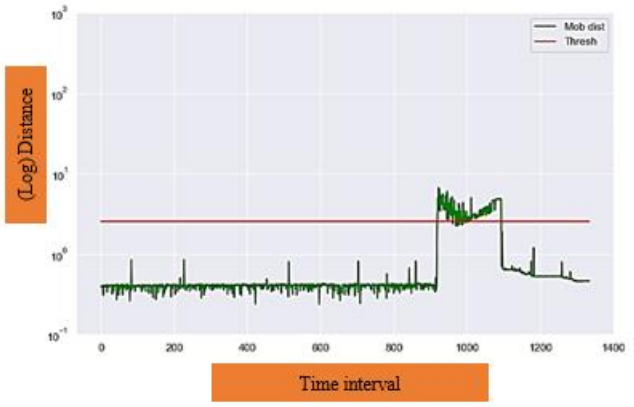

Figure 13. Plot of Mahalanobis distance vs Threshold

Table. 1. Performance Evaluation Results

Window Size	**8**
True Negatives (TN)	1805
True Positives (TP)	149
False Negatives (FN)	32
False Positives (FP)	7
True Positive Rate (TPR)	0.8232
False Positive Ratio (FPR)	0.0039
Positive Predictive Value (PPV)	0.9551
True Negative Rate (TNR)	0.9961
Negative Predictive Value (NPV)	0.9826
False Negative Rate (FNR)	0.1768
ACC	0.9804
F1 score	0.8843

Table 1 depicts the numerical (Machine Learning) ML-based result metrics for our ad-hoc intrusion detection technique (sliding window). It is, therefore, clearly apprehended that the (Sliding Window) SW technology appears to outrun, since a well-standing, but totally versatile approach (PCA/Mahalanobis distance), seems equivalent to it in terms of overall precision, or detection accuracy. We depict our

numerical finding results in Table 1 following an empirical size for the (sliding) window. We comprehend the crucial importance of early detection for any types of security anomalies within IOTA-based infrastructures; and our technique manages to achieve that aim in a sound and robust manner. In an overall manner, our IOTA intrusion detection system achieves **an accuracy level of 98% (ACC), with an F-measure of 88%**.

A final discussion point for the readers would be why we (pre)select a static window size of 8 for the sliding window ad-hoc approach? Our numerical findings appear to *empirically* explain the case. It seems fair enough that this empirical value outruns far better than less, or much higher values. One potential reason is that we have four resource monitoring metrics (e.g., ***CPU usage, memory usage, minor faults, and major faults***), thus the ideal constant declaration for such window size would be double the amount of the metric inputted to the ML-Classifier. Still, we leave the same discussion for future work, in terms of increasing the number or monitoring metrics; should the window size adjust?

5 Future Work Recommendations

In future we plan to refine our approach by studying the impact of additional parameters, using the data generated for Denial of Service (DoS) attacks. In addition, we plan to include additional parameters and distribute weights based on the parameter importance extracted from data mining techniques. Our work only considered zero-value transactions; the experiments can be extended by studying the impact of non-zero value transactions compared to zero value transactions to the IOTA network from a DoS attack perspective. Future work can be improved by introducing a reinforcement learning (RL) rule into the sliding window, to increase anomaly detection accuracy. In the future, we plan to apply regressor to predict threat index, in addition to detecting the threat index. This will lead to a peak performance of threat index accuracy.

6 Conclusion

As per literature review, the integration of DLT to IoT makes the IoT devices more vulnerable and prone to attacks. One of the important attacks that we addressed is DoS attacks. We implemented the threat model and our security algorithms in the practical DLT-based private tangle network. In our research, we demonstrated the potential of our security algorithms and methods to protect the DLT-based IoT system and integrate our methods to the existing lightweight DLT framework for IoT. The proposed approach could be potentially used to securely deploy blockchain framework- based IoT and low powered peer to peer systems in the real-world applications.

References

1. Danzi, P, Kalor, A. E, Stefanovic, C and Popovski. P: Analysis of the Communication Traffic for Blockchain Synchronization of IoT Devices, 2018 IEEE International Conference on Communications (ICC), pp. 1-7, IEEE Press, New York (2001).
2. Varshney, G and H. Gupta. : A security framework for IoT devices against wireless threats 2017 2nd International Conference on Telecommunication and Networks (TEL-NET), pp. 1-6, , IEEE Press, New York (2017).
3. Ghiro, L, Maccari, L and Cigno, R.L: Proof of networking: Can blockchains boost the next generation of distributed networks?, 2018 14th Annual Conference on Wireless On-demand Network Systems and Services (WONS), Isola, pp.29-32 (2018)
4. Kumar, S.A, Vealey, T. and Srivastava.H.: Security in Internet of Things: Challenges, Solutions, and Future Directions. In 49th Hawaii International Conference on System Sciences (HICSS), pp. 5772-5781, IEEE Press, New York (2016).
5. Bikos, A. N., and Sklavos, N. : The Future of Privacy and Trust on the Internet of Things (IoT) for Healthcare: Concepts, Challenges, and Security Threat Mitigations. In Recent Advances in Security, Privacy, and Trust for Internet of Things (IoT) and Cyber-Physical Systems (CPS) (pp. 63-90). Chapman and Hall/CRC. (2020)
6. Nakamoto, S.: ”Bitcoin: A peer-to-peer electronic cash system,” 2008.
7. Christidis, K and Devetsikiotis, M: Blockchains and Smart Contracts for the Internet of Things. IEEE Access, vol. 4 (2016).
8. Atzori, L, Iera, A. Morabito, G.: The Internet of Things: A survey, In Computer Networks, Volume 54, Issue 15, pp. 2787-2805, (2010).
9. Roy, S, Ashaduzzaman, M, Hassan, M and Chowdhury, A. R..: BlockChain for IoT Security and Management: Current Prospects, Challenges, and Future Directions. in 5th International Conference on Networking, Systems and Security (NSysS), 2018, pp. 1-9, (2018).
10. Muzammal, S. M and Murugesan, R.K.: A Study on Leveraging Blockchain Technology for IoT Security Enhancement. in Fourth International Conference on Advances in Computing, Communication & Automation (ICACCA), pp. 1-6 (2018).
11. Yin, S, Bao, J, Zhang, Y and Huang, X.: M2m security technology of cps based on blockchains. Symmetry, vol. 9, no. 9, p. 193 (2017).
12. Rahulamathavan, Y, Phan, R. Phan, C.W, Rajarajan, M, Misra, S and Kondoz, A.: Privacy-preserving blockchain-based IoT ecosystem using attribute-based encryption. in 2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS). pp. 1–6. IEEE Press, New York (2017).
13. Hammi, M. T, Hammi, B, Bellot, P and Serhrouchni, A.: Bubbles of trust: A decentralized blockchain-based authentication system for iot. Computers & Security, vol. 78, pp. 126–142. (2018).
14. B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, ”Blockchain based data integrity service framework for IoT data,” in Web Services (ICWS), 2017 IEEE International Conference on. IEEE, 2017, pp. 468–475.
15. Bramas, Q.: The Stability and the Security of the Tangle. hal-01716111v2 (2018).
16. Popov, S.: The Tangle. white paper. [Online]. Available: https://iota.org/ (2016)
17. Heilman, E., Narula, N., Tanzer, .G, Lovejoy, J., Colavita, M., Virza, M., and Dryja. T.: Cryptanalysis of Curl-P and Other Attacks on the IOTA Cryptocurrency. IACR Cryptology ePrint Archive : 344 (2019).
18. Roode, D., Gerard, I.U., and Havinga, P. J: How to break IOTA heart by replaying?. In 2018 IEEE Globecom Workshops (GC Wkshps), pp. 1-7. IEEE Press, New York (2018)
19. Tennant, L. : Improving the Anonymity of the IOTA Cryptocurrency. (2017)

20. Moubarak, J, Chamoun, M., and Filiol. E: On distributed ledgers security and illegal uses. Future Generation Computer Systems 113: 183-195. (2020).
21. Tekeoglu, A., and Ahmed, N. :TangoChain: A Lightweight Distributed Ledger for Internet of Things Devices in Smart Cities. In 2019 IEEE International Smart Cities Conference (ISC2): Blockchain Enabled Sustainable Smart Cities (Bless 2019) Workshop (IEEE ISC2 2019-Bless Workshop), (2019)
22. Fernández-Caramés, T.M., and Fraga-Lamas, P. :A review on the use of blockchain for the internet of things. IEEE Access, vol. 6,pp. 32 979–33 001 (2018)
23 Lin, I.C and Liao, T.C.: A survey of blockchain security issues and challenges. Network Security, vol. 19, no. 5, pp. 653–659. (2017)
24. Atzei, N, Bartoletti, M and Cimoli, T.: A survey of attacks on Ethereum smart contracts sok. in Proceedings of the 6th International Conference on Principles of Security and Trust, vol. 10204, pp. 164–186. (2017)
25.Siegel, D.: Understanding The DAO Attack. https://www.coindesk.com/understanding-dao-hack-journalists/.
26. C. Baldwin, "Bitcoin worth 72 million stolen from bitfinex exchange in Hong Kong," http://reut.rs/2gc7iQ9, Aug 2016, accessed online, Nov 25, 2019.
27. Khatibzadeh, L., Bornaee, Z., and Bafghi, A.: Applying Catastrophe Theory for Network Anomaly Detection in Cloud Computing Traffic. Security and Communication Networks. 1-11. (2019)
28. Robust covariance estimation and Mahalanobis distances relevance. https://scikit-learn.org/stable/auto_examples/covariance/plot_mahalanobis_distances.html.
29. Ester, M., Kriegel H., Sander, J, and Xu, X : A density-based algorithm for discovering clusters in large spat ial databases with noise. In: Proceedings of the second internat ional conference on knowledge discovery and data mining (KDD-96), pp 226–231, (1996)
30 Bikos, A.N., and Kumar, S.: Securing Digital Ledger Technologies-Enabled IoT Devices: Taxonomy, Challenges and Solutions. IEEE Access. (2022)
31. Alampalayam, S.K., Kumar, A., and S. Srinivasan. :Statistical Based Intrusion Detection Framework using Six Sigma Technique. IJCSNS 7, no. 10: 333 (2007).
32. Alampalayam, S.P., and Kumar, A: Security model for routing attacks in mobile ad hoc networks. In 2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No. 03CH37484), vol. 3, pp. 2122-2126. IEEE Press, New York, (2003).
33. Alampalayam, S.K., and Natsheh, E.F: Multivariate fuzzy analysis for Mobile ad hoc Network threat Detection. International Journal of Business Data Communications and Networking (IJBDCN) 4, no. 3, pp:1-30 (2008)
34. Srinivasan, S., and S. P. Alampalayam. :Intrusion detection algorithm for MANET. International Journal of Information Security and Privacy (IJISP) 5, no. 3, pp. 36-49 (2011)
35. Gohil, M., and Kumar, S. :Evaluation of classification algorithms for distributed denial of service attack detection. in 2020 IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), pp. 138-141. IEEE Press, New York (2020).
36. Bikos, A. N., and Kumar, S. :Reinforcement learning-based anomaly detection for Internet of Things distributed ledger technology." in 2021 IEEE Symposium on Computers and Communications (ISCC), pp. 1-7. IEEE Press New York, (2021).