IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received February 14, 2017, accepted March 9, 2017, date of publication April 24, 2017, date of current version September 27, 2019.

Digital Object Identifier 10.1109/ACCESS.2017.2690456

A Novel Internet of Things-Centric Framework to Mine Malicious Frequent Patterns

NIGHAT USMAN[1], QAISAR JAVAID[2], ADNAN AKHUNZADA[1], KIM-KWANG RAYMOND CHOO[4], (Senior Member, IEEE), SAEEDA USMAN[3], ASMA SHER[1], MANZOOR ILAHI[1], AND MASOOM ALAM[1]

[1]Department of Computer Science, COMSATS Institute of Information Technology, Islamabad, Pakistan
[2]Department of Computer Science & Software Engineering, International Islamic University, Islamabad, Pakistan
[3]Department of Electrical Engineering, COMSATS Institute of Information Technology, Sahiwal, Pakistan
[4]Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249-0631, USA

Corresponding author: A. Akhunzada (a.adnan@siswa.um.edu.my)

ABSTRACT There are a number of research challenges associated with Internet of Things (IoT) security, and one of these challenges is to design novel frameworks to mine malicious frequent patterns for identifying misuse and detecting anomalies without incurring high computational costs (e.g., due to generation and analysis of unnecessary patterns and gap creation between patterns). Association rule mining is a popular approach in the literature; hence, in this paper, we critically analyze existing association rule mining techniques. We then present a framework for mining malicious frequent patterns in an IoT deployment, prior to evaluating the utility of the proposed framework using data from a Pakistan-based organization.

INDEX TERMS Malicious behavior, security logs, Internet of Things (IoTs), frequent pattern mining, anomaly detection.

I. INTRODUCTION

Internet of Things (IoT) is a recent promising trend, which has widespread applications. Specifically, in an IoT implementation, device to device communications [1] can be achieved using communication platforms such as wireless sensor networks, radio frequency identification, Bluetooth, etc. The amount of data store and transfer between IoT devices in a typical deployment is increasing, partly due to the digitization of our society [2]–[4]. Construction of IoTs has progressed fundamentally in last couple of years – see Table 1.

This, however, results in vectors that could potentially be exploited by attackers to infiltrate a system and/or exfilitrate data from IoT devices. as billions of devices are interconnected, so networks are vulnerable to a range of security threats which in turns have an adverse effect on performance of the thing [6], [7]. Not surprisingly, IoT security and privacy have attracted the attention of both researchers and policy makers.

A popular area of research is using data mining techniques, such as classification, association analysis, statistical learning, link mining and clustering, to analyze and discover relationships, patterns and other useful information from IoT raw data [8]–[10]. Association rule mining (ARM) is one such popular technique, which allows one to determine the correlations among all features of a dataset, and detect anomalies [11]. ARM has been used in a number of applications, such as market basket analysis whose objective is to identify / predict trends (commonly grouped items) within the data [12]. In the latter context, findings are used to determine which items are more probable to be paired with other items to make an itemset [13], [14].

ARM technique comprises of multiple algorithms such as Apriori, FP-Growth association rule, Prefixspan, Spade, and Spam. For example, the Apriori algorithm was proposed in 1994 by Agarwal and Srikant. This level-wise bottom-up approach is used to extract frequent itemsets (candidate generation) from a dataset. It finds the itemsets according to the specified minimum support count. However, there are some limitations with Apriori. For instance, multiple scans are required. An explicit scan is required for each candidate set that can lead to an I/O cost. Moreover, all required patterns are not guaranteed. The computational cost is also high due to the need for large storage and processing time. Pattern fragment growth is another method commonly used to mine the complete set of frequent patterns (FP-Growth Association Rule Mining). This method uses a divide-and-conquer approach to create a relationship among multiple items. The processing speed is comparatively fast and it utilizes the space

TABLE 1. Device connections and expected growth: A snapshot [5].

S. No.	Organization	Connected Devices Forecast
01	CISCO Projects	37 Billion Interconnected Objects by 2020
02	ABI Research	Over 30 billion wireless connected devices by 2020
03	Gartner	25 billion IoTs devices by 2020
04	Business Insider	23.4 billion IoTs connected devices by 2019
05	Harbor Projects	21.7 billion IoTs connected devices by 2019
06	Navigant Research	Roughly 7.2 billion "M2M connected consumer electronic devices" by 2023
07	IHS Automotive	Connected cars to the Internet will be 152 million by 2020
08	On World	Roughly 100 million wireless interconnected lights by 2020

in a better way. However, this method is inefficient when the dataset has a large number of items and the patterns are twinned. Prefixspan (i.e., Prefix-projected Sequential pattern mining) mines the complete set of patterns in sequential pattern mining. Unlike FP-Growth, the attempts of candidate sequel generation are significantly reduced. It uses the divide and conquer approach to discover hidden patterns in the database [15]. The limitations with Prefixspan include the processing required for further child possible patterns and Gaps.

These algorithms, however, do not perform well in network security applications, as we would need to locate malicious patterns in the IoT real-time traffic [16]. For example, due to high costs in memory, resources for processing, creation of unnecessary pattern and gap creation between the patterns, these algorithms are not suitable for suspicious pattern mining – see also Section V.

In order to overcome the limitations in these algorithms, the Mine Malicious Frequent Patterns (MMFP) algorithm is proposed in this paper. MMFP is designed to mine frequent patterns efficiently to facilitate the detection of anomalous behavior. However, prior to presenting MMFP in Section III, we will first discuss related work (see Section II).

II. RELATED WORK

The process to find hidden behavior in a dataset that may either be benign or abnormal, is known as anomaly detection. While anomaly refers to an unexpected behavior, not all anomalies are cyber or attempted attacks [17]–[20]. Anomaly detection has widespread applications such as detecting fraudulent credit card transactions and malicious network activities [21]–[25]. For instance, a behavior profile of a legitimate user is learned by the algorithm, and any behavior that deviates from the typical behavior (e.g. a banking transaction in a different country or conducted when the user is supposed to be asleep) will trigger an alarm [26]–[28].

A number of data mining techniques designed to detect malicious activity have been proposed in the literature. For example, techniques such as Prefixspan [29], Apriori [30], GSP [31] and FP-Growth [32] are used to locate hidden patterns and their associations. This allows the identification of zero day attacks. Agrawal *et al.* [33] proposed the AIS algorithm, which uses the previous acquaintance of the recursive item set. In this algorithm, level-wise search is used. In order to search kitems, we need to have (k-1) itemsets, and in order to find recursive 1itemsets, the database is scanned to obtain counts for each item. Any item that satisfies the minimum support will produce a result labeled L1. Recursive 2itemsets i.e. L2 is found by using L1. The process continues until no k-itemset can be found. However, this requires the searching of the entire database for each itemset. To reduce the search efforts, Apriori property is used. According to this property, every non-empty subset of recursive itemset must also be recursive [34].

To improve search efficiency, another method was introduced to remove the itemsets that may not be frequent. By doing so, counting is reduced for these itemsets. For efficient memory management, one could remove redundant itemsets or to remove larger set having parent and children and store only the parent on the disk at the start of the next iteration [35]. A limitation with AIS is that a lot of candidate itemsets are generated which are not useful due to their small size. Also, the database is scanned multiple times, which results in unnecessary time and CPU cycles [36].

The *Apriori* algorithm is proposed for ARM in [30]. Apriori is comparatively accurate and faster than AIS. In Apriori, there are two steps involved in discovering the large-size itemsets. First, the candidate sets are created. To find the support threshold value, the database is scanned. Next, we prune itemsets whose frequencies are less than the pre-defined support threshold value. Apriori avoids investigating candidate itemsets that are rare [37]. Consequently, after pruning again and again, the leftover candidate sets are reduced. Therefore, the requirements for I/O, computational cost and memory are also slightly reduced [38]. However, Apriori still scans the entire databases several times and does not guarantee all required patterns due to limitation in storage.

To find frequent items, the *MCAR* (Multi-class Classification based on Association Rule) is proposed. MCAR consists of a rule generation, and a building classifier. In rule generation, MCAR studies the training dataset to identify the frequent k items. After that, it recursively pools the items to return items that have more attributes. Then, ranks are created and the patterns of these itemsets stored. At the second stage, rules are defined for the patterns that were stored to build a classifier [39]. Accuracy and speed are two important performance parameters. The advantages of using MCAR is the ability to identify frequent items and rules in only single pass; thus, saving on storage and execution time. By following a method of rule ranking, a random choice to pick one rule among different rules is minimized [40].

In *RARM* (Rapid Association Rule Mining) [41], the database is viewed in the form of a tree rather than a candidate generation process. RARM is much quicker than FP-Tree. In the sizable itemsets generation process, there are two phases:

i) *Preprocessing phase:* SOTrieIT (Support Ordered Trie Itemset) structure is used to rapidly generate big size 1-itemsets and 2-itemsets from every transaction. By doing so, scanning the database and candidate generation for the next time is not required. Similar to FP-Tree, every link of SOTRie IT bears one item and the relevant support count. SOTrieIT is the enhanced version of TrieIT. TrielIT is similar to SOTrieIT. However, in TrieIT, more memory is required due to the need for individual storage of support counts. SOTrieIT is introduced to reduce the storage requirement.

ii) *Mining large itemsets:* by following a depth-first approach, SOTrieIT tree first scans the leftmost first level node and checks for the minimal support threshold value at each level. After the generation of big size 1-itemsets and 2-itemsets, the Apriori algorithm is applied to identify some new large itemsets.

Although the most expensive operation during the process of mining is to create the biggest size 2 itemsets. However, findings in [41] demonstrate that generation of big size 1-itemsets and 2-itemsets through SOTrieIT algorithm can be improved. However, SOTrieIT has the same limitations to FP-Tree [36], [42].

FP-Tree (Frequent Pattern Tree) [32] is an ARM technique that mines better than Apriori as it overcomes two limitations of Apriori. Rules are generated using a tree structure of multiple items. By scanning only the database twice, frequent itemsets are generated without generating a candidate set. Thus, the databases are scanned only twice and it is much faster than Apriori [43]. This involves two child processes:

i) *Building FP-Tree:* Similar to the Apriori algorithm, it first scans the database, collects the support count of all items, sorts the frequent itemsets in descending order by considering their support values. By doing so, sequential frequent 1-itemsets is generated.

ii) *Creating frequent patterns using FP-Tree:* An FP-Tree is constructed by scanning the database again with the main table. For each iteration, frequent items' state is re-sorted according to the main table. For instance; the T1 (I2, I4, I6) is changed to T2 (I4, I2, I6) because I4 occurs more frequently than I2 in the dataset.

Frequent patterns use a divide-and-conquer approach. While this has a low computational cost since no candidate set is generated, the FP-Tree is not suitable for incremental mining and interactive mining system as in the incremental mining approach, databases continually change as time passes. This is because records may be updated or newly inserted, and this updating may results in repetition of the entire process. While in interactive mining system, legitimate users can alter the minimum support threshold value by considering the rules and it too results in repetition of the entire process. This approach also generates similar patterns of itemsets, according to the frequency [36], [44].

A comparative summary of AIS, Apriori, MCAR, RARM, and FP-Growth is presented in Table 2. and GSP, SPADE, SPAM, and Prefixspan is presented in Table 3.

The *GSP* (Generalized Sequential Pattern) algorithm [31] is an effective method to examine ordered patterns which uses the bottom-up approach. However, by decreasing the minimum support value, large number of candidates are generated. Thus, this requires significant time and resources [45], [46]. Candidate k-sequences are generated from (k-1)-sequences. Depending on the support count, the candidate k-sequence frequency is obtained in every iteration [47]. To address this over generated candidate set problem, the authors proposed using the SPADE algorithm to split the candidate sequences into blocks [48]. A bottom-up approach is used in SPADE to obtain the regular sequences [49]. In order to reduce the cost, an ID-List technique is used to compute support count. This ID-List keeps a record of pairs that indexes the positions in the database. However, a single sequence can be recorded more than once. SPADE is costly when the number of candidate sequences is large and when continual merging of Id-lists are required [50]. To reduce merging costs, the authors in [51] proposed the *SPAM* algorithm where every ID-list is viewed as a straight icon. As all icons can be stored in the RAM, the algorithm has a fast performance [52], [53].

Prefixspan is designed to mine frequent items from a dataset, using a divide-and-conquer approach to discover hidden patterns in the database. Here, unlike FP-growth, the number of candidate sequel generation is greatly reduced. In order to discover the frequent 1-sequences, such as <(x)>, <(y)>, <(z)>, <(a)>, <(b)>, it scans the database first. Next, projected database is generated for all frequent 1-sequence [29]. To detect the frequent (k+1)-sequences, the Prefixspan algorithm recursively creates the projecting databases for every frequent k-sequence. However, similar to SPADE and SPAM, Prefixspan is costly and extra Gapsİ are created while yielding sizable projected databases twinned patterns are generated [53]–[55].

As previously discussed, ARM techniques have been widely used in networking context. For instance, the authors in [56] demonstrated the utility of association rules in extracting intrusion patterns from tcpdump log file and system call logs. In [57], the author identifies events of interest from the MAWI traffic depository using frequent itemset mining technique in traces [58]. Chandola and Kumar [59] outlined heuristic program rules for discovering small set of frequent itemsets that can sum up sizeable sets of flows. The "eXpose" [60] is an application that exploits the impermanent correlation between flows in a very small time stamp window and detects abnormal communication patterns [61].

TABLE 2. A comparative summary of existing data mining approaches: AIS, Apriori, MCAR, RARM, and FP-growth.

	AIS	Apriori	MCAR	RARM	FP-Growth
Processing time	large	large >FP-Growth	less	less than FP-Growth	less
Cost	very high	high	high	less	less
Strategy	BFS	BFS	rule ranking review approaches	DFS	divide and conquer and DFS hipp
Accuracy	less	less	more	more	more
Memory	inefficient	efficient>AIS	efficient	efiicient	inefficient
Iterations	multiple passes	lower >AIS	single pass	two passes	two passes
Pros	easy to use	fast,generates candidate sets from only large items and compresses data set	minimize the randomization process for rules	avoid to generatecandidate set	only 2 passes of dataset and no candidate set generation required
Cons	generates candidate sets on-the-fly and large size of candidate set	large storage required and scans the whole DB multiple times	uses only one rule for predicting test cases	tree structure creates complexity	tree structure creates complexity and inefficient for interactive and incremental mining

TABLE 3. A comparative summary of existing data mining approaches: GSP, SPADE, SPAM, and Prefixspan.

	GSP	SPADE	SPAM	Prefixspan
Processing time	large	large >GSP	less	high
Cost	very high	less	very high	high
Strategy	BFS	DFS	depth first search	divide and conquer and DFS
Accuracy	less than FP-Growth	more	less than SPAM	less than FP-Growth
Memory	same as Prefixspan	efficient	less efficient than SPADE	efficient
Iterations	multiple passes	threepasses	multiple passes	single pass
Pros	discovering generalized sequential structure,	fast,reduce I/O cost by reducing DB scans	both algorithm and data structure stored in main memory	no need to generatecandidate set
Cons	large candidate sequences and redundant patterns	takes very large running time	requires more space	make a projected db for every sequential pattern

In the proposed framework, we have evaluated performance of Prefixspan, FP-Growth and Apriori mining techniques. We have compared the performance against our novel approach (as mentioned in section V. All aforementioned shortcomings affiliated to these three techniques are resolved in the proposed framework. We claim that Prefixspan generates surplus gaps; however, our algorithm efficiently overcome this issue. Prefixspan and FP-Growth generates redundant patterns due to which large storage is utilized and performance is compromised. Likewise, FP-Growth, Apriori is also not preferable for large datasets as all required patterns are not guaranteed, and thus it requires extra storage and surplus processing time which is unfordable in IoT(s). We have overcome these deficiencies by converting all records to the same data type through mask creation (as mentioned in section 3.B), due to which processing time, power and storage utilization is reduced [62]. Our proposed scheme MMFP, expeditiously results in exact existing patterns with no gaps. Hence, an optimized framework is launched that can deliver better performance for secure communication among things. The resulting patterns from this structure aid firms to discover anomalies and respond accurately against them in things to things communication.

III. PROPOSED FRAMEWORK FOR MINING MALICIOUS FREQUENT PATTERN

Our proposed scheme is illustrated in Fig. 1, which consists of three key phases.

A. FEATURE SELECTION

Determining the right dataset for pattern mining can be challenging but this is an important step, as the choice of accurate features for dataset will dictate the effectiveness of discovering anomalies. In our proposed framework, the essential features are extracted using Principle Component Analysis (PCA). It is used to investigate and conceptualize the data by concentrating on fluctuation(s) in the dataset. Features having identical values are removed and those with most variations are selected. This phase is described in Algorithm 1.

Algorithm 1 Feature Selection

Input: *dataset.txt*
Output: *legends_dataset.txt* ▷ A file containing records with legends.
procedure
  for each $l_i \in logs$ **do**
    extract redundant features
    apply PCA ▷ Principle Component Analysis.
    store in *feature_extr.txt*
  endfor
  if features extraction done **then**
    call **Legends Creation**
  endif
  return *legends_dataset.txt*

B. LEGENDS CREATION

In practice, it is challenging to build the desired dataset manually out of log files as there are various types of log files with different kinds of attributes. Moreover, the features of the dataset also need to be managed properly, as we have to set appropriate data types of the features in order to prepare them for association. In order to make the dataset appropriate for computations, we have to convert the string values to numerical values.

0 $\longrightarrow$ feature should not be selected for dataset;
1 $\longrightarrow$ feature should be selected the way it is;
2 $\longrightarrow$ feature should be selected and assigned with legends.

After the selection of attributes using PCA, the next step is to create legends by making mask. Mask is the structure that facilitates the construction of legends of a specific data type. In order to make our scheme space efficient, non-numeric values can be converted to numeric values using mask, which ranges between [0-2] according to the requirement of values of the attributes. With a view to convert all values to the same data type, a novel approach is proposed and demonstrated in Algorithm 2.

The desired datasets can be formed through a mask as it is user-defined. After converting values to a single data type,

Algorithm 2 Legends Creation

Input:
a) *feature_extr.txt*
b) *mask.txt*
Output: *legends_dataset.txt* ▷ A file containing records with legends.
procedure
  for each $M_i \in Mask$ **do**
    if $M_i == 0$ **then**
      move to next
    endif
    else
    if $M_i == 1$ **then**
      store M_i in *legends_dataset.txt*
      M++
    endif
    else
    if $M_i == 2$ **then**
      $M_i \leftarrow UL$ ▷ A file containing records with legends.
      create *UL.attr.txt*
      write $M_i \mid UL$
      store UL in *legends_dataset.txt*
    endif
  endfor
  return *legends_dataset.txt*

Algorithm 3 Main Framework

Input: *legends_dataset.txt*
Output: *OFP.txt*
procedure
  for each $F_i \in legend$ **do**
    calculate index of F_i
    $I_f \leftarrow indexofF_i$
    $j \leftarrow I_f + 1$
    while line(j) != F$_i$ **do**
      j++
    end while
    for $k \leftarrow I_f; k < j$ **do**
      for $z \leftarrow I_f - 1; z < k$ **do**
        write line[z] in the OFP.txt
        move to next line
      endfor
    endfor
  endfor
  return *OFP.txt*

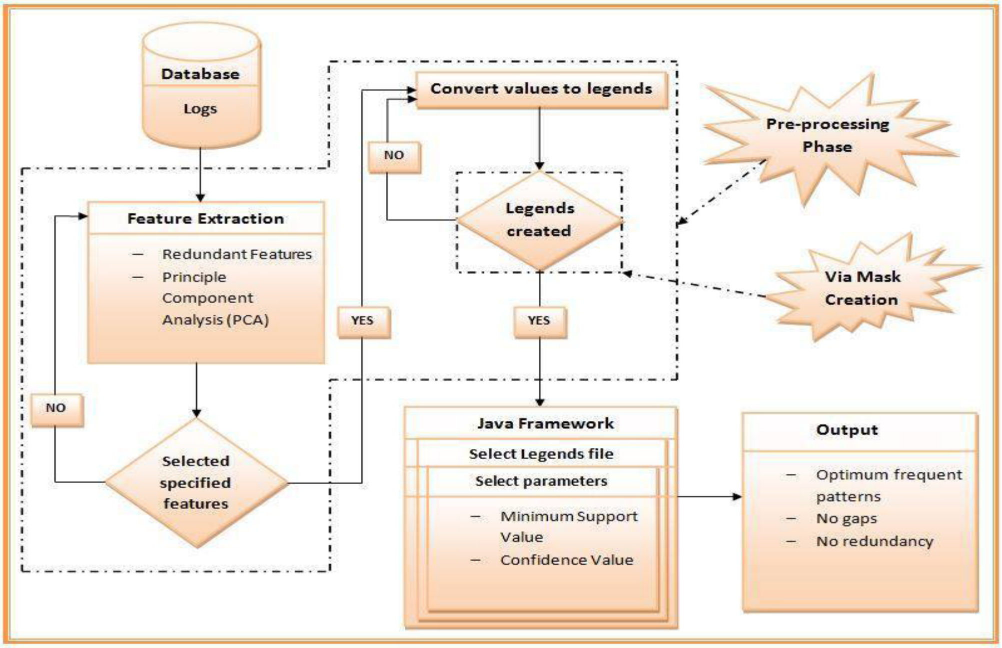

FIGURE 1. Proposed framework for mining malicious frequent patterns (MMFP).

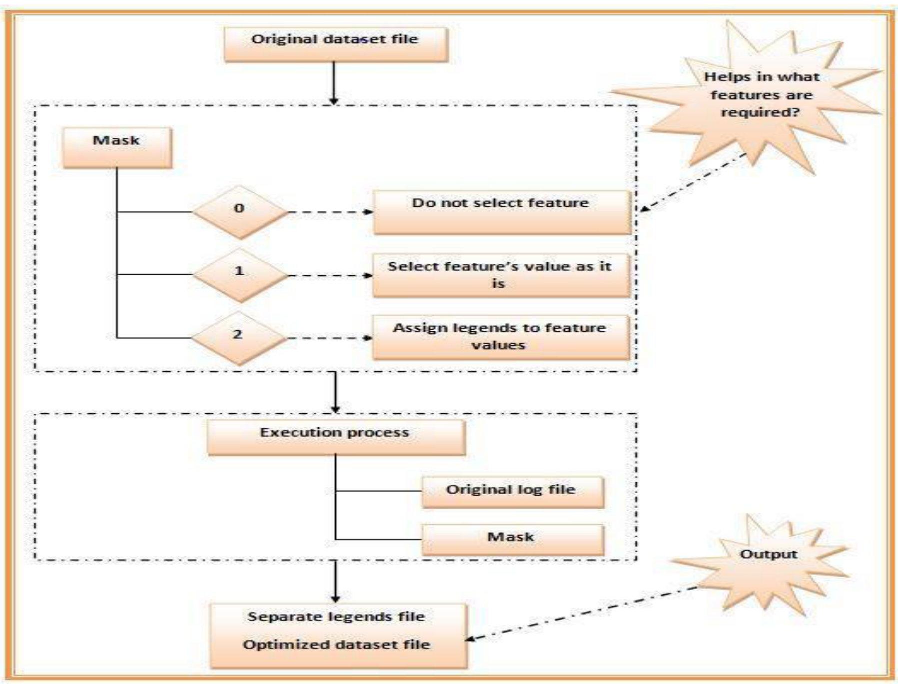

FIGURE 2. Framework for mask creation.

the scheme consumes less memory; thus, more cost efficient. Fig. 2 outlines the complete legend assignment phase.

C. FREQUENT PATTERN MINING

Existing algorithms do not only result in redundant patterns, but also yield extra gaps; thus, consuming more memory and incurring additional costs. MMFP mines frequent patterns without producing extra gaps and redundant patterns. The main framework of the proposed approach is described in Algorithm 3.

IV. PARAMETERS SELECTION

Two user-defined parameters, that are incorporated by the proposed algorithm which are listed below:

TABLE 4. Comparison of ARM techniques and MMFP for a sample of 50 records.

	Transactions	Min. Support Value	Min. Confi-dence Level	Max. Memory Usage	Candidates Count	Frequent Itemsets Count	Time	Association Rules Generated	Rules Generation Time
Apriori	50	0.05 = 3	0.3	4.34mb	1380	66	48ms	22	16ms
Prefixspan	50	0.06 = 3	–	13.49mb	–	113	152ms	24118	26s
FP-Growth	50	0.03 = 3	0.3	51.64mb	–	168699	452ms	41042460	265s
MMFP	50	0.03 = 3	0.3	4.12mb	–	34	19ms	50	19ms

TABLE 5. Comparison of ARM techniques and MMFP for a sample of 100 records.

	Transactions	Min. Support Value	Min. Confi-dence Level	Max. Memory Usage	Candidates Count	Frequent Itemsets Count	Time	Association Rules Generated	Rules Generation Time
Apriori	100	0.03 = 3	0.26	14.67mb	2856	110	49ms	61	25ms
Prefixspan	100	0.03 = 3	–	14.06mb	–	165	274ms	39387	29s
FP-Growth	100	0.02 = 3	0.3	113.05mb	–	376563	568ms	87324837	804s
MMFP	100	0.027 = 3	0.3	4.97mb	–	65	28ms	100	21ms

TABLE 6. Comparison of ARM techniques and MMFP for a sample of 150 records.

	Transactions	Min. Support Value	Min. Confi-dence Level	Max. Memory Usage	Candidates Count	Frequent Itemsets Count	Time	Association Rules Generated	Rules Generation Time
Apriori	150	0.02 = 3	0.26	15.5mb	4290	148	72ms	107	22ms
Prefixspan	150	0.02 = 3	–	14.116mb	–	204	388ms	54959	35s
FP-Growth	150	0.02 = 3	0.3	145.31mb	–	569097	672ms	13430716	2434s
MMFP	150	0.02 = 3	0.3	6.68mb	–	120	41ms	150	24ms

TABLE 7. Comparison of ARM techniques and MMFP for a sample of 200 records.

	Transac-tions	Min. Support Value	Min. Confi-dence Level	Max. Memory Usage	Candidates Count	Frequent Itemsets Count	Time	Association Rules Generated	Rules Generation Time
Apriori	200	0.014 = 3	0.26	15.35mb	6343	184	103ms	129	26ms
Prefixspan	200	0.015 = 3	–	14.34mb	–	234	419ms	73138	49s
FP-Growth	200	0.01 = 3	0.3	202.25mb	–	921194	1833ms	19952584	3950s
MMFP	200	0.02 = 3	0.3	9.05mb	–	174	63ms	200	29ms

- **Minimum Support Value (MSV):** In order to collect frequent patterns that occur more than the specified threshold value, the Minimum Support Value parameter is used. MSV takes a value in [0-1].
- **Confidence Level:** For observing the occurrence of a pattern in the dataset, a confidence level is specified which takes a value in [0-1].

The MMFP algorithm scans the entire dataset at once to locate frequent patterns according to MSV and confidence level; thus, avoiding limitations in existing algorithms. Unlike traditional algorithms, all frequent patterns that reside in the original database become the output file. The algorithm discards non-frequent patterns and ensures an optimum solution. After the frequent patterns have been mined, rules are created for every resultant pattern. If a pattern deviates from normal, then expert analysis is performed on that sequence to determine whether it is an anomaly.

V. FINDINGS AND DISCUSSIONS

We now present a comparative summary of Apriori, Prefixspan, FP-Growth and the proposed framework MMFP. In the evaluations, we used a machine of at least 6GB RAM, and 2.10Giga Hertz processor. A sample of 200 security pertinent records from a dataset of 10,48,569 records of a real-world case study organization (Trillium Pakistan) was segregated in 4 chunks (i.e. 50, 100, 150 and 200), and the performance metrics used were processing time, memory utilization, and number of generated patterns.

Table 4 displays the outcomes of using Apriori, Prefixspan, FP-Growth and MMFP for 50 records. Based on the

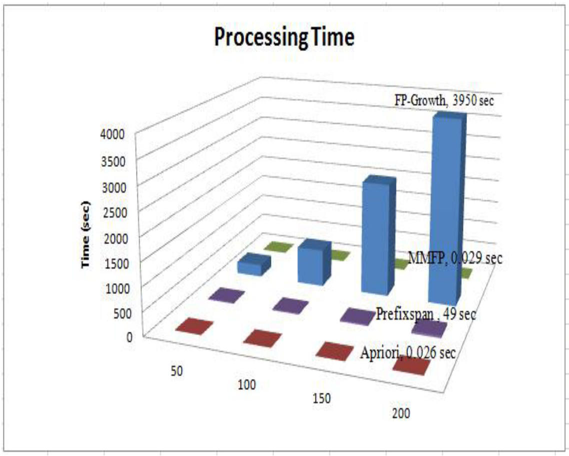

FIGURE 3. Processing time for multiple transactions.

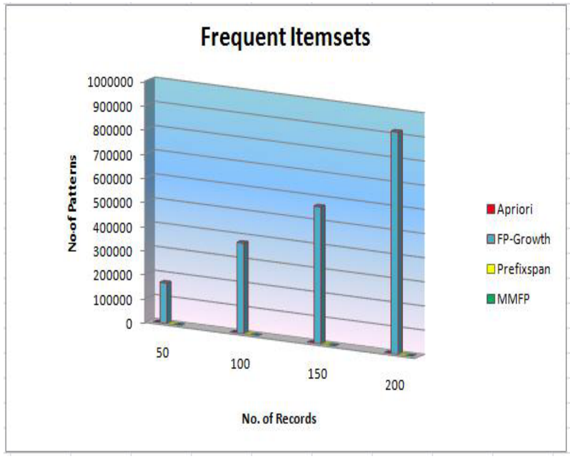

FIGURE 4. Frequent patterns generated for multiple transactions.

specified value for number of transactions and minSup, the algorithms were then evaluated. By comparing the results, we observed that MMFP performs comparatively well with respect to memory usage. Prefixspan needs more memory as it creates the projecting databases for every frequent k-sequence and generates extra gaps in the patterns. Therefore, it requires additional time for rules generation and is not suitable for environment that consists of resource-constrained devices (e.g. IoT devices) [63]. In terms of memory usage and time, FP-Growth performance is very poor because of redundant pattern generation. Table 5 displays the outcomes of using Apriori, Prefixspan, FP-Growth and MMFP for 100 records. MMFP outperforms Apriori in terms of memory usage.

The other findings for 150 and 200 records are reported in Tables 6 and 7, respectively. As observed from Table 7.

From our evaluations, we observed that Apriori does not work well for large dataset as it does not guarantee all frequent patterns. Prefixspan modifies existing frequent patterns by creating gaps in order to generate every possible pattern; however, these patterns are of little use in detecting anomalies. FP-Growth is also not suitable for large datasets as it generates identical patterns more than once; thus, incurring additional costs. The processing time for multiple transactions of the four algorithms is illustrated in Fig. 3. It is clear that FP-Growth results in the most number of frequent itemsets as it returns surplus patterns. Prefixspan with a large difference shows frequent itemsets, and Apriori achieves better performance than FP-Growth and Prefixspan in all segregated portions. MMFP executes all transactions with the optimal number of frequent itemsets and thus, results in optimal use of memory and processing time.

Fig. 4 depicts the rule generation processing time for all the four algorithms, it is clear that MMFP has the best processing time.

From Figs. 5.a), 5.b), 5.c) and 5.d), we observed that MMFP requires less memory compared to the three algorithms as it generates meaningful but fewer number of

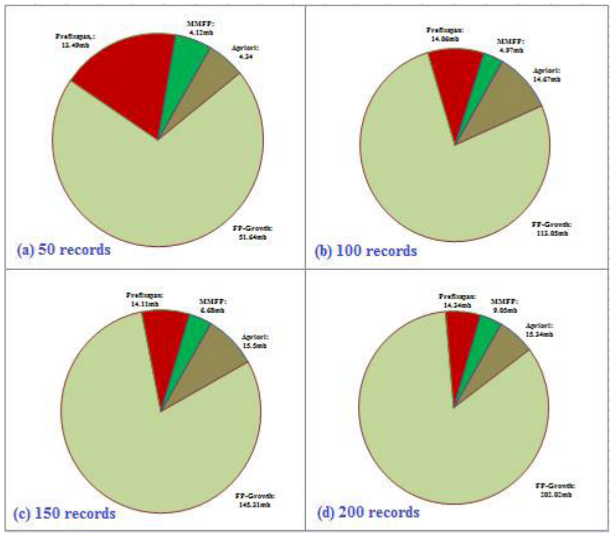

FIGURE 5. Memory required in multiple transactions.

patterns.

The products and systems that connect to the IoT are changing business in many commercial enterprises. MMFP can be used anywhere such as in scheduling maintenance for auto vehicles, predicting results through E-voting, tracking exact record for home appliances, medical applications for patients to capture their health data and so on.

VI. CONCLUDING REMARKS

Ensuring the security of data and devices in an IoT infrastructure can be challenging due to the nature of such infrastructure (e.g. resource-constrained devices). Thus, it is generally accepted that deployed security solutions should be lightweight [64], [65].

In this paper, we presented a framework to mine malicious frequent patterns in IoT communications that would allow us to identify misuse and detect anomalies without incurring high computational costs. We evaluated the utility of our proposed framework by benchmarking the framework with three popular ARM techniques, namely: Apriori, FP-Growth and Prefixspan, using real-world security logs from a Pakistan-based organization.

Future research includes deploying the proposed framework on multiple datasets and systems to further fune-tune the framework for enhanced efficiency and accuracy. In order to make a proficient recommendation system, there is a need to associate MMFP with Deep Learning techniques for inter-communication and intra-communication among Internet of Things (IoTs).

REFERENCES

[1] Q. Alam *et al.*, "Formal verification of the xDAuth protocol," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 9, pp. 1956–1969, Sep. 2016.

[2] D. Quick and K.-K. R. Choo, "Big forensic data management in heterogeneous distributed systems: Quick analysis of multimedia forensic data," *Softw., Pract. Exper.*, to be published.

[3] D. Quick and K.-K. R. Choo, "Pervasive social networking forensics: Intelligence and evidence from mobile device extracts," *J. Netw. Comput. Appl.*, to be published.

[4] D. Quick and K.-K. R. Choo, "Digital forensic intelligence: Data subsets and Open Source Intelligence (DFINT+OSINT): A timely and cohesive mix," *Future Generat. Comput. Syst.*, to be published.

[5] A. D. Thierer, "The connected world: Examining the Internet of Things," Tech. Rep. SSRN 2563765, 2015.

[6] D. Singh, G. Tripathi, and A. J. Jara, "A survey of Internet-of-Things: Future vision, architecture, challenges and services," in *Proc. IEEE World Forum Internet Things (WF-IoT)*, Mar. 2014, pp. 287–292.

[7] C. J. D'Orazio, K.-K. R. Choo, and L. T. Yang, "Data exfiltration from Internet of Things devices: iOS devices as case studies," *IEEE Internet Things J.*, to be published.

[8] R. Alur *et al.* (2016). "Systems computing challenges in the Internet of Things." [Online]. Available: https://arxiv.org/abs/1604.02980

[9] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining*. 1996.

[10] B. Thuraisingham, "A primer for understanding and applying data mining," *IT Prof.*, vol. 2, no. 1, pp. 28–31, Jan. 2000.

[11] T. Steen and R. Lindsay, "RecB: Set theory based technique for large scale pattern mining in Web logs," *Int. J. Comput. Appl.*, vol. 124, no. 8, pp. 1–9, 2015.

[12] C. C. Aggarwal, *Data Mining: The Textbook*. Springer, 2015.

[13] J. Han and Y. Fu, "Mining multiple-level association rules in large databases," *IEEE Trans. Knowl. Data Eng.*, vol. 11, no. 5, pp. 798–805, Sep. 1999.

[14] M.-S. Chen, J. Han, and P. S. Yu, "Data mining: An overview from a database perspective," *IEEE Trans. Knowl. Data Eng.*, vol. 8, no. 6, pp. 866–883, Dec. 1996.

[15] A. Guevara-Cogorno, C. Flamand, and H. Alatrista-Salas, "COPPER—Constraint optimized prefixspan for epidemiological research," *Procedia Comput. Sci.*, vol. 63, pp. 433–438, 2015.

[16] M. Sookhak *et al.*, "Remote data auditing in cloud computing environments: A survey, taxonomy, and open issues," *ACM Comput. Surv.*, vol. 47, no. 4, p. 65, 2015.

[17] J. West and M. Bhattacharya, "Intelligent financial fraud detection: A comprehensive review," *Comput. Secur.*, vol. 57, pp. 47–66, Mar. 2016.

[18] A. Akhunzada *et al.*, "Secure and dependable software defined networks," *J. Netw. Comput. Appl.*, vol. 61, pp. 199–221, Feb. 2015.

[19] A. Akhunzada *et al.*, "Man-at-the-end attacks: Analysis, taxonomy, human aspects, motivation and future directions," *J. Netw. Comput. Appl.*, vol. 48, pp. 44–57, Feb. 2015.

[20] B. Arrington, L. Barnett, R. Rufus, and A. Esterline, "Behavioral modeling intrusion detection system (BMIDS) using Internet of Things (IoT) behavior-based anomaly detection via immunity-inspired algorithms," in *Proc. 25th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Aug. 2016, pp. 1–6.

[21] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, p. 15, 2009.

[22] M. E. Edge and P. R. F. Sampaio, "A survey of signature based methods for financial fraud detection," *Comput. Secur.*, vol. 28, no. 6, pp. 381–394, 2009.

[23] J. Peng, K.-K. R. Choo, and H. Ashman, "User profiling in intrusion detection: A review," *J. Netw. Comput. Appl.*, vol. 72, pp. 14–27, Sep. 2016.

[24] S. Iqbal *et al.*, "On cloud security attacks: A taxonomy and intrusion detection and prevention as a service," *J. Netw. Comput. Appl.*, vol. 74, pp. 98–120, Oct. 2016.

[25] P. Dokas, L. Ertoz, V. Kumar, A. Lazarevic, J. Srivastava, and P.-N. Tan, "Data mining for network intrusion detection," in *Proc. NSF Workshop Next Generat. Data Mining*, 2002, pp. 21–30.

[26] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Comput. Sci.*, vol. 60, pp. 708–713, 2015.

[27] R. J. Baxley, C. J. Rouland, and M. T. Engle, "Anomalous behavior detection based on behavioral signatures," U.S. Patent 2015 0 350 233, Dec. 3, 2015.

[28] H. Banuri *et al.*, "An android runtime security policy enforcement framework," *Pers. Ubiquitous Comput.*, vol. 16, no. 6, pp. 631–641, 2012.

[29] Y. Xu and Y. Wang, "Analysis of Web access sequence based on the improved PrefixSpan algorithm," in *Proc. Int. Ind. Inform. Comput. Eng. Conf. (IIICEC)*, 2015, pp. 788–791.

[30] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. 20th Int. Conf. Very Large Data Bases (VLDB)*, vol. 1215. 1994, pp. 487–499.

[31] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *Advances in Database Technology*. Springer, 1996.

[32] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 1–12, 2000.

[33] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," *ACM SIGMOD Rec.*, vol. 22, no. 2, pp. 207–216, 1993.

[34] K. Khurana and S. Sharma, "A comparative analysis of association rules mining algorithms," *Int. J. Sci. Res. Pub.*, vol. 3, no. 5, pp. 1–4, 2013.

[35] L. M. Goyal, M. M. S. Beg, and T. Ahmad, "A novel pruning approach for association rule mining," *BVICAM's Int. J. Inf. Technol.*, vol. 7, no. 1, pp. 827–834, 2015.

[36] J. C.-Y. N. Xian-Jun, "Association rule mining: A survey," *Comput. Sci.*, vol. 4, p. 044, 2003.

[37] D. Adhikary and S. Roy, "Trends in quantitative association rule mining techniques," in *Proc. IEEE 2nd Int. Conf. Recent Trends Inf. Syst. (ReTIS)*, Jul. 2015, pp. 126–131.

[38] S. Rao and P. Gupta, "Implementing improved algorithm over APRIORI data mining association rule algorithm 1," Tech. Rep., 2012.

[39] W. Hadi, "EMCAR: Expert multi class based on association rule," *Int. J. Modern Edu. Comput. Sci.*, vol. 5, no. 3, pp. 33–41, 2013.

[40] F. Thabtah, P. Cowling, and Y. Peng, "MCAR: Multi-class classification based on association rule," in *Proc. 3rd ACS/IEEE Int. Conf. Comput. Syst. Appl.*, Jan. 2005, p. 33.

[41] A. Das, W.-K. Ng, and Y.-K. Woon, "Rapid association rule mining," in *Proc. 10th Int. Conf. Inf. Knowl. Manage.*, 2001, pp. 474–481.

[42] Ö. M. Soysal, "Association rule mining with mostly associated sequential patterns," *Expert Syst. Appl.*, vol. 42, no. 5, pp. 2582–2592, 2015.

[43] Y. Zeng, S. Yin, J. Liu, and M. Zhang, "Research of improved fp-growth algorithm in association rules mining," *Sci. Program.*, vol. 2015, Jan. 2015, Art. no. 910281.

[44] Z. Rong, D. Xia, and Z. Zhang, "Complex statistical analysis of big data: Implementation and application of Apriori and FP-Growth algorithm based on MapReduce," in *Proc. 4th IEEE Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, May 2013, pp. 968–972.

[45] Y. Fan, Y. Ye, and L. Chen, "Malicious sequential pattern mining for automatic malware detection," *Expert Syst. Appl.*, vol. 52, pp. 16–25, Jun. 2016.

[46] X. Cheng, S. Su, S. Xu, P. Tang, and Z. Li, "Differentially private maximal frequent sequence mining," *Comput. Secur.*, vol. 55, pp. 175–192, Nov. 2015.

[47] M.-T. Tran, B. Le, B. Vo, and T.-P. Hong, "Mining non-redundant sequential rules with dynamic bit vectors and pruning techniques," *Appl. Intell.*, vol. 45, no. 2, pp. 333–342, 2016.

[48] M. J. Zaki, "SPADE: An efficient algorithm for mining frequent sequences," *Mach. Learn.*, vol. 42, nos. 1–2, pp. 31–60, 2001.

[49] A. P. Wright, A. T. Wright, A. B. McCoy, and D. F. Sittig, "The use of sequential pattern mining to predict next prescribed medications," *J. Biomed. Inform.*, vol. 53, pp. 73–80, Feb. 2015.

[50] C. A. Fowler and R. J. Hammell, "Mining information assurance data with a hybrid intelligence/multi-agent system," in *Proc. IEEE/ACIS 14th Int. Conf. Comput. Inf. Sci. (ICIS)*, Jun./Jul. 2015, pp. 23–28.

[51] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, "Sequential pattern mining using a bitmap representation," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2002, pp. 429–435.

[52] C. Liu, X. Dong, C. Li, and Y. Li, "SAPNSP: Select actionable positive and negative sequential patterns based on a contribution metric," in *Proc. 12th Int. Conf. Fuzzy Syst. Knowl. Discovery (FSKD)*, Aug. 2015, pp. 811–815.

[53] M. Verma, D. Mehta, V. Dahiya, and K. Mehta, "Mining sequences—Approaches and analysis," *Int. J. Innov. Res. Sci. Technol.*, vol. 1, no. 7, pp. 229–233, 2015.

[54] D.-Y. Chiu, Y.-H. Wu, and A. L. P. Chen, "An efficient algorithm for mining frequent sequences by a new strategy without support counting," in *Proc. IEEE 20th Int. Conf. Data Eng.*, Apr. 2004, pp. 375–386.

[55] K. K. Arya, V. Goyal, S. B. Navathe, and S. Prasad, "Mining frequent spatial-textual sequence patterns," in *Database Systems for Advanced Applications*. Springer, 2015, pp. 123–138.

[56] W. Lee and S. J. Stolfo, "Data mining approaches for intrusion detection," in *Proc. Usenix Secur.*, 1998.

[57] K. Yoshida, Y. Shomura, and Y. Watanabe, "Visualizing network status," in *Proc. IEEE Int. Conf. Mach. Learn.*, vol. 4. Aug. 2007, pp. 2094–2099.

[58] The MAWI Working Group of the WIDE Project, "MAWI working group traffic archive," Tech. Rep., 2012.

[59] V. Chandola and V. Kumar, "Summarization—compressing data into an informative representation," *Knowl. Inf. Syst.*, vol. 12, no. 3, pp. 355–378, 2007.

[60] S. Kandula, R. Chandra, and D. Katabi, "What's going on?: Learning communication rules in edge networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 87–98, 2008.

[61] M. Alam *et al.*, "Dynamic remote attestation through behavior measurement and verification," *Int. J. Innov. Comput. Inf. Control*, vol. 8, no. 3(A), pp. 1821–1836, 2012.

[62] M. Alam *et al.*, "Optimizing SIEM throughput on the cloud using parallelization," *PLoS ONE*, vol. 11, no. 11, p. e0162746, 2016.

[63] Q. Alam, S. U. Malik, A. Akhunzada, K.-K. R. Choo, S. Tabbasum, and M. Alam, "A cross tenant access control (CTAC) model for cloud computing: Formal specification and verification," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 6, pp. 1259–1268, Jun. 2016.

[64] Y. Yang, H. Cai, Z. Wei, H. Lu, and K.-K. R. Choo, "Towards lightweight anonymous entity authentication for IoT applications," in *Information Security and Privacy* (Lecture Notes in Computer Science), vol. 9722. 2016, pp. 265–280.

[65] Y. Yang, J. Lu, K.-K. R. Choo, and J. K. Liu, "On lightweight security enforcement in cyber-physical systems," in *Lightweight Cryptography for Security and Privacy* (Lecture Notes in Computer Science), vol. 9542. 2015, pp. 97–112.

NIGHAT USMAN, photograph and biography not available at the time of publication.

QAISAR JAVAID, photograph and biography not available at the time of publication.

ADNAN AKHUNZADA, photograph and biography not available at the time of publication.

KIM-KWANG RAYMOND CHOO, photograph and biography not available at the time of publication.

SAEEDA USMAN, photograph and biography not available at the time of publication.

ASMA SHER, photograph and biography not available at the time of publication.

MANZOOR ILAHI, photograph and biography not available at the time of publication.

MASOOM ALAM, photograph and biography not available at the time of publication.