J Ambient Intell Human Comput (2018) 9:1141–1152
DOI 10.1007/s12652-017-0558-5

ORIGINAL RESEARCH

Detecting crypto-ransomware in IoT networks based on energy consumption footprint

Amin Azmoodeh[1] · Ali Dehghantanha[2] · Mauro Conti[3] · Kim-Kwang Raymond Choo[4]

Received: 5 March 2017 / Accepted: 28 July 2017 / Published online: 23 August 2017

Abstract An Internet of Things (IoT) architecture generally consists of a wide range of Internet-connected devices or things such as Android devices, and devices that have more computational capabilities (e.g., storage capacities) are likely to be targeted by ransomware authors. In this paper, we present a machine learning based approach to detect ransomware attacks by monitoring power consumption of Android devices. Specifically, our proposed method monitors the energy consumption patterns of different processes to classify ransomware from non-malicious applications. We then demonstrate that our proposed approach outperforms K-Nearest Neighbors, Neural Networks, Support Vector Machine and Random Forest, in terms of accuracy rate, recall rate, precision rate and F-measure.

✉ Ali Dehghantanha
a.dehghantanha@salford.ac.uk

Amin Azmoodeh
azmoodeh@cse.shirazu.ac.ir

Mauro Conti
conti@math.unipd.it

Kim-Kwang Raymond Choo
raymond.choo@fulbrightmail.org

[1] Department of Computer Science and Engineering, Shiraz University, Shiraz, Iran

[2] Department of Computer Science School of Computing Science and Engineering, University of Salford, Greater Manchester, UK

[3] Department of Mathematics, University of Padua, Padua, Italy

[4] Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX, USA

Keywords Ransomware detection · Power consumption · Internet of Things security · Machine learning · Malware detection · Android

1 Introduction

The Internet of Things (IoT) refers to interrelated connected network of smart devices, sensors, embedded computers and etc. that store,process and communicate heterogeneous data. IoT and its applications propagate to majority of life's infrastructure ranging from health and food production to smart cities and urban management. While efficiency and prevalence of IoT are increasing, security issues remain a necessary concern for industries (Tankard 2015). Internet-connected devices, including those deployed in an IoT architecture, are increasingly targeted by cybercriminals due to their pervasiveness and the ability to use the compromised devices to further attack the underlying architecture (Choo 2014; Pajouh et al. 2016; D'Orazio et al. 2017; Fortino and Trunfio 2014; Watson and Dehghantanha 2016). In the case of ransomware, for example, devices that are capable of storing a reasonably amount of data (e.g., Android and iOS devices) are likely to be targeted (Damshenas et al. 2015; D'Orazio and Choo 2016; Gubbi et al. 2013). Thus, ensuring the security of IoT nodes against threats such as malware is a topic of ongoing interest (Bertino et al. 2016; Sicari et al. 2015; Kumar and Patel 2014; Abomhara and Kien 2015; Daryabar et al. 2012; Teing et al. 2017; Dezfouli et al. 2016).

While malware detection and mitigation research is now new, ransomware detection and mitigation remains challenging. Ransomware is a relatively new malware type that attempts to encrypt a compromised device's data using a strong encryption algorithm (O'Gorman and McDonald 2012). The victim will then have to pay the ransom (usually

using bitcoins) in order to obtain the password or decryption key (Song et al. 2016). Consequences include temporary or permanent loss of sensitive information, disruption of regular operations, direct/indirect financial losses (e.g., to restore systems and restore an organizations reputation) (FBI 2016).

A popular malware detection approach is the use of machine learning techniques to identify patterns of specific feature(s) within a malware code or behavior to distinguish malware from non-malicious applications (Faruki et al. 2015; Damshenas et al. 2015; Pajouh et al. 2016). For example, Andronio et al. (2015) proposed an Android ransomware detection system, Heldroid, that is based on Natural Language Processing (NLP). The approaches identify ransomware based on their typical characteristics, such as call function and application manifests. EldeRan (Sgandurra et al. 2016) is another machine learning based model for dynamically analyzing and classifying ransomware based on their installation activities. Mercaldo et al. (2016) presented a parser that analyzes a sample code and automatically identifies ransomware related instructions. In Caviglione et al. (2016), malware covert communications are detected using neural networks and decision tree techniques.

Changes in the energy consumption of a typical infected device can also be used as a feature for malware detection (Caviglione et al. 2016), as it could be trivial for a malware developer to change malware function calls or its behaviour but changing its power usage pattern is less likely and more difficult to realise (Shaerpour et al. 2013). Additionally, power usage pattern is relatively similar on different platforms; thus, power consumption based detection methods appear to be a viable approach (Potlapally et al. 2006). Kim et al. (2008) proposed a power-aware malware detection framework based on anomalies in a device energy consumption pattern. Similarly, Merlo et al. (2015) demonstrated the potential of detecting a malware on an Android device based on its energy consumption.

In this paper, we use machine learning techniques to detect ransomware based on their power usage patterns on IoT nodes, and specifically Android devices. The proposed model grinds device's power usage into subsamples, classifies them and aggregates outputs to increase the detection rate to 95.65%.

2 Related literature

Securing IoT nodes is an active research area. For example, Sicari et al. (2015) and Jing et al. (2014) discussed several key research challenges and identified potential solutions and research opportunities for IoT security, and Abomhara and Kien (2015) provided a categorisation of IoT related threats.

Malware detection and mitigation for IoT nodes is one of several research challenges and opportunities identified, and is an ongoing research topic (Faruki et al. 2015; Suarez-Tangil et al. 2014). Detection methods include those based on malware's properties (e.g., application signatures), and tracking of malicious activity and their energy consumption (Shaerpour et al. 2013). Malware detection based on energy consumption footprint is known to be more robust against malware anti-forensic techniques as changing a malware power consumption pattern is much more challenging in practice, compared to changing its function calls or application codes (Damshenas et al. 2013).

Kim et al. (2008) proposed a power-aware malware detection framework that detects previously unknown battery-draining malware. Their framework comprises a power monitoring tool and a data analyzer which generates a power signature to identify a malware. Merlo et al. (2015) presented an energy-related measurement at a different levels of abstraction for Android devices in order to achieve a trade-off between measurement precision and effective energy based profiling of malware. Yang and Tang (2016) used the frequencies of energy consumption waveform to generate a Gaussian Mixture Model (GMM) based on Mel frequency cepstral coefficients (MFCC) to detect malicious software. In Yang and Tang (2016), authors use a complex statistical approach to make decision based on power usage. This, however, is generally too computationally expensive for IoT nodes. The authors also employed frequencies of waveform in their approach; therefore, changes in the CPU's specification would have a substantial impact on the results even though the waveform's visual form remains invariant.

Machine learning algorithm, as previously discussed, has been widely employed in cyber security research, including malware detection. Andronio et al. (2015) presented *Heldroid* to detect Android ransomware *Heldroid* based on file encryption activities using a NLP-based text classifier, locking detector and a tracker. *Heldroid* utilises extracted features from malware application such as alert messages, function call and etc. Sgandurra et al. (2016) proposed *EldeRan* for dynamically analysing and classifying ransomware based on the set of actions performed by the applications in their installation phase. *EldeRan* is designed for Windows platform and the most relevant feature to the class label is *Registry Keys Operations*, which is not applicable for Android devices. Mercaldo et al. (2016) presented a three-step process to detect Android ransomware family. Similar to other static malware detection approaches, techniques such as code metamorphism could be used to evade detection.

Table 1 Ransomware

MD5 hash	Download date
30a03d7a5e6ec234bbb6d333e9f30ec9	14 Oct 2016
597bbb81e6409a389299aa8ded222e8b	5 Oct 2016
6315c783974743327f8d19c67c465f28	13 Oct 2016
37cd3ac4d5acda83a5512032c99ea279	12 Oct 2016
e1b9eb7415892ef6ca3fda9f304428a6	12 Oct 2016
902c4044dc7872382001e2e3e36a8c0f	11 Oct 2016

3 Research methodology

To develop a fingerprint of ransomware's energy consumption, initially, we need to record the power usage of targeted applications. Similar to the approaches in previous studies (Yang 2012; Merlo et al. 2015) we used *PowerTutor* to monitor and sample power usage of all running processes in 500 ms intervals. PowerTutor creates logfiles containing sequence of energy usage of each process at given sampling interval. We conducted our experiments on three different Android devices, namely: a Samsung Galaxy SIII (CPU: 1.4 GHz, RAM: 2GB, OS: Android 4.4), a Samsung Galaxy S Duos (CPU: 1.0 GHz, RAM: 768 MB, OS: Android 4.0.1), and an Asus Padfone Infinity (CPU: 1.7 GHz, RAM: 2 GB, OS: Android 4.4). To collect energy consumption logs of both ransomware and goodware, we installed the most popular Android applications, namely: *Gmail (version 9.6.83), Facebook (version 99.0.0.26.69), Google Chrome (version 53.0.2785.124), Youtube (version 11.39.56), Whatsapp (version 2.16.306), Skype (version 7.20.0.411), AngryBrids (version 6.1.5), Google Maps (version 9.39.2), Music Player (version 4.2.52), Twitter (version 6.19.0), Instagram (version 9.6.0) and Guardian (version 3.13.107)* and six active and recent ransomware samples (see Table 1) on all devices. All ransomware were downloaded via VirusTotal[1] Intelligence API, and these ransomware have active Command and Control (C2) servers.

We then use *PowerTutor* to monitor and record the device processes' power usage (while running the applications and ransomware, separately) for 5 min. While running the applications (also referred to as goodware), the user interactions mirrored a real world usage. This procedure was repeated five times per device; thus, we obtained $5_{repeation} \times 3_{device} = 15$ power usage samples for each and every application and ransomware.

As each device's CPU has its own power usage specification, the energy consumption of all devices were mapped to a specific range in order to have a meaningful evaluation. So, we normalised the CPU power consumption for all monitored processes on the devices to [0, 1], where 0 indicates no power usage and 1 presents the maximum CPU power utilisation. Scripts were written to process logfiles, extract and normalize power usage values, and generate a row-normalized dataset. Each row includes a label (i.e., goodware or ransomware) and a normalized sequence of energy consumption for five minutes of activity.

3.1 Classification

Assigning correct label to a sample based on previous observations is a key element of Supervised Learning and Classification (Michalski et al. 2013). We applied four state-of-the-art classifiers, namely: k-Nearest Neighbor (KNN), Neural Network (NN), Support Vector Machine (SVM) and Random Forest (RF), on the power usage samples to recognise the class of each sequence of power consumption. KNN is a simple and powerful classifier which seeks K nearest sample(s) and assigns the majority of neighbor's label to the given samples (Cover and Hart 1967). NN (Haykin 1998) is an implementation of human brain networks and mostly used to approximate the function between inputs and output. Another popular technique for supervised learning is SVM (Burges 1998), which is based on the concept of decision planes that define decision boundaries. A decision plane differentiates a set of objects based on their class memberships. Ensemble learning has been the motivation of developing RF (Verikas et al. 2011) that operates by constructing a multitude of decision trees at training time and generating the class label.

Power usage sequence of each process can be considered as time-series data. A wide range of methods have been proposed to classify time-series data (Xing et al. 2010; Fu 2011). In this study, a distance based time-series classification approach based on Dynamic Time Warping (DTW) (Müller 2007) is used for distance measure, and KNN is used as a classifier. Similarity distance is a key element in KNN classification and we apply two different distances to find the closest neighbor as follows:

- Euclidean distance: Euclidean distance or Euclidean metric is the intuitive distance between two vectors in Euclidean space and calculated as follow:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \quad (1)$$

- Dynamic time warping (DTW): DTW is a recognized technique for finding an optimal alignment between two time-dependent sequences (see Fig. 1). According to DTW's ability to deal with time deformations and issues associated with speed differences in time-dependent

[1] http://www.virustotal.com.

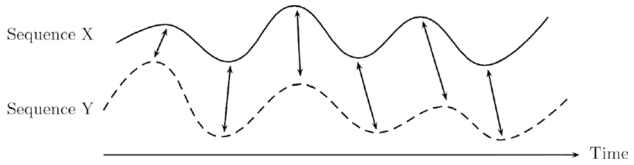

Fig. 1 Alignment between two different sequences

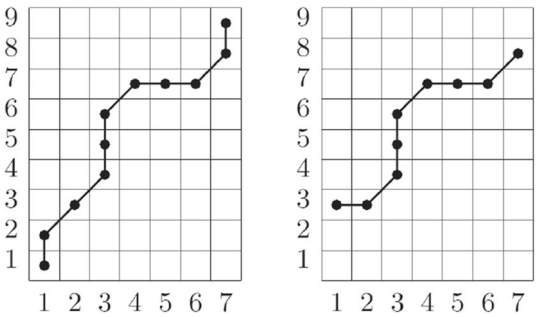

Fig. 2 Dynamic Time Warping's path finding

data, it is also employed to calculate distance or similarity between time series (Müller 2007). Let us denote two sequences that display two discrete subsamples as $X = (x_1, \ldots, x_n)$ and Y $= (y_1, \ldots, y_m)$ of length $m, n \, \epsilon \, \mathbb{N}$. DTW uses a *Cost Matrix* $C \epsilon \mathbb{R}^{n \times m}$. Each cell $C_{i,j}$ indicates the distance between x_i and y_j (see Fig. 2). DTW's purpose is to discover an optimal alignment between X and Y having a minimal entirely distance. As an intuitive explanation, an optimal alignment traverse across a valley of low cost cells within the cost matrix C. A warping path is specified as a sequence $p = \{p_1, \ldots, p_L\}$ with $p_l = (n_l, m_l) \epsilon [1{:}N] \times [1{:}M], l \epsilon [1{:}L]$ satisfying the following conditions:

- Boundary condition: $p_1 = (1, 1)$ *and* $p_L = (N, M)$.
- Monotonicity condition: $n_1 \leq n_2 \leq \cdots \leq n_L$ *and* $m_1 \leq m_2 \leq \cdots \leq m_L$.
- Step size condition: $p_{l+1} - p_l = \{(1, 0), (0, 1), (1, 1)\}$ *for* $l \epsilon [1{:}L1]$.

The summation of all local distances of a warping path's elements outcomes the total cost of path and in order to find *optimal warping path* p^*, the path having minimum total cost among all possible paths is selected. Finally, to measure similarity or distance between two sequences X and Y, their total cost of *optimal warping path* are evaluated. The total cost $c_p(X, Y)$ of a warping path p between X and Y with respect to the local cost measure c is defined as:

$$c_p(X, Y) = \sum_{l=1}^{L} c(x_{nl}, y_{ml}). \tag{2}$$

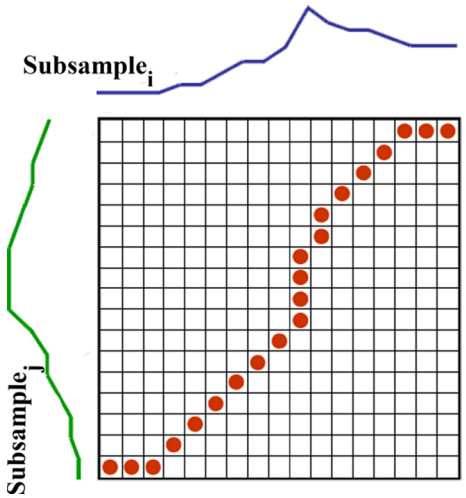

Fig. 3 Dynamic time Warping's path finding example for power usage sequences

The DTW distance $DTW(X, Y)$ between X and Y is then defined as the total cost of p^*:

$$DTW(X, Y) = c_{p*}(X, Y) = min\{c_p(X, Y) | \; p \; is \; an \; (N, M) - warping \; path\}. \tag{3}$$

Figure 3 illustrates how *DTW* aligns two power usage subsamples in order to find optimal path between them for distance calculation.

3.2 Metrics and cross-validation

Similar to the approach in (Buczak and Guven 2016), we use the following four common performance indicators for malware detection:

- True positive (TP): indicates that a ransomware is correctly predicted as a malicious application.
- True negative (TN): indicates that a goodware is detected as a non-malicious application correctly.
- False positive (FP): indicates that a goodware is mistakenly detected as a malicious application.
- False negative (FN): indicates that a ransomware is not detected and labelled as a non-malicious application.

To evaluate the effectiveness of our proposed method, we used machine learning performance evaluation metrics that are commonly used in the literature, namely: Accuracy, Recall, Precision and F-Measure.

Accuracy is the number of samples that a classifier correctly detects, divided by the number of all ransomware and goodware applications:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{4}$$

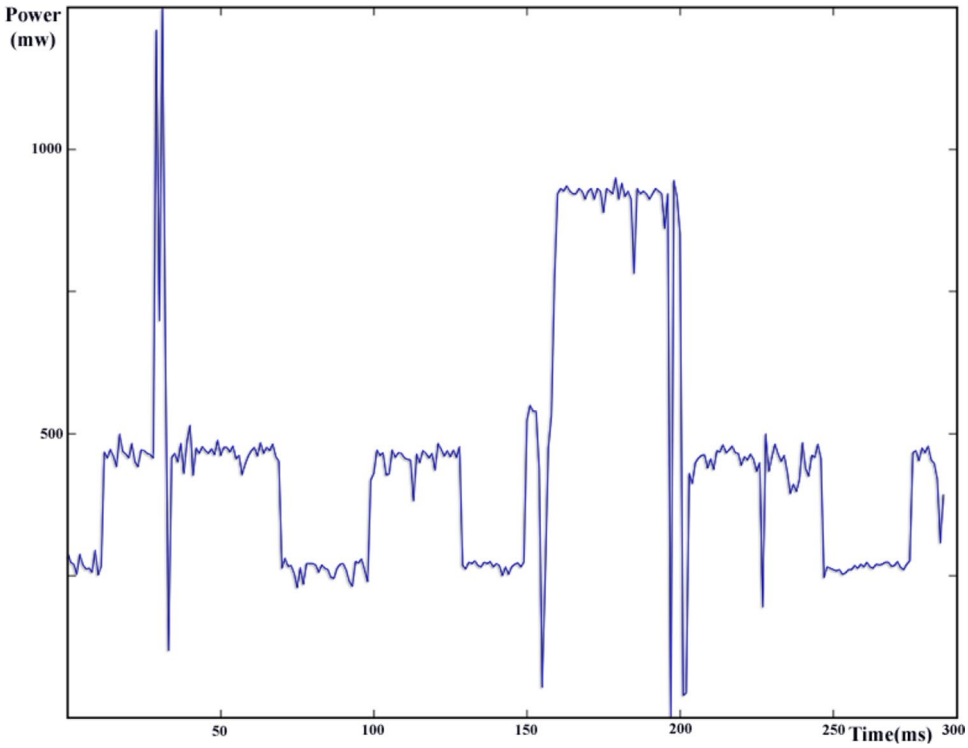

Fig. 4 Power consumption graph for Simplocker ransomware

Precision is the ratio of predicted ransomware that are correctly labelled a malware. Thus, Precision is defined as follows:

$$Precision = \frac{TP}{TP + FP}. \tag{5}$$

Recall or detection rate is the ratio of ransomware samples that are correctly predicted, and is defined as follows:

$$Recall = \frac{TP}{TP + FN}. \tag{6}$$

F-Measure is the harmonic mean of precision and recall, and is defined as follows:

$$F - Measure = \frac{2 * TP}{2 * TP + FP + FN}. \tag{7}$$

Cross-validation (Kohavi et al. 1995) is a fundamental technique in machine learning to assess the extent that the findings of an experiment can be generalized into an independent dataset. In order to evaluate the performance of the proposed method, we used the leave-one-out cross validation. We are aware that in order to implement this validation method, all subsamples of a sample need to be excluded

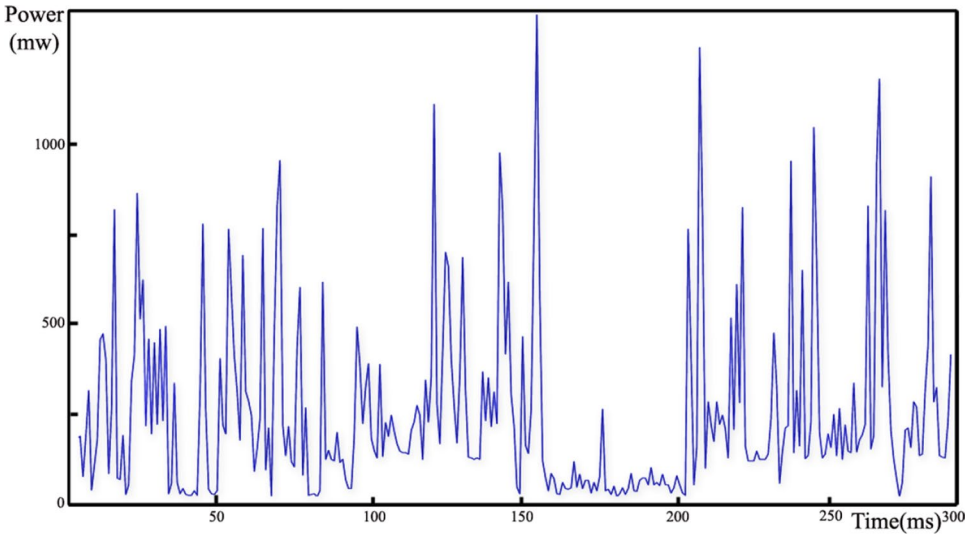

Fig. 5 Power consumption graph for facebook application

Table 2 Performance of machine learning techniques: a comparative summary

	Accuracy (%)	Recall (%)	Precision (%)	F-Measure (%)
KNN (K = 1)	71.85	71.11	56.14	62.75
KNN (K = 5)	72.59	72.22	57.02	63.73
KNN (K = 10)	72.22	71.11	56.64	63.05
KNN (K = 1 and DTW)	**83.70**	**78.89**	**73.96**	**76.34**
Neural network	75.93	73.33	61.68	67.01
Random forest	80.74	76.67	69.00	72.63
SVM	78.52	74.44	65.69	69.79

Best (optimal) values are highlighted in bold

from the classifier training phase. All evaluations were conducted using MATLAB R2015a running on a Microsoft Windows 10 Pro personal computer powered by Intel Core i7 2.67 GHz and 8 GB RAM.

We will evaluate the performance of the classification algorithms in the next section.

4 Performance of classification algorithms

Table 2 displays the findings of applying classification algorithms on our dataset. As previously discussed, we will now use the leave-one-out technique for cross validation. Figures 4 and 5 illustrate the power usage graph of Simplocker ransomware and Facebook application, respectively.

Comparison of Figs. 5 and 4 reflects a significant difference between patterns of power consumption of ransomware versus benign applications. However, as patterns of power consumptions are not predictable and depend on many factors such as files content, encryption algorithm etc. samples are highly distributed in the feature space.

It appears that direct application of conventional classification algorithms namely NN, KNN and SVM, is not promising. For example, the KNN classifier that uses DTW as a similarity measure outperformed other techniques while conventional KNN (with parameter setting of K = 1, 5, 10) is ranked lowest among the classification approaches.

Since Euclidean method calculates similarity by summing distances between corresponding points of samples, the calculated distance could be far when the position of occurring power usage patterns varies (even if samples are visually cognate). On the other hand, DTW attempts to align samples based on the distance between pieces of samples that are more similar regardless of the position of similar energy usage pattern. Consequently, the performance of KNN classifier is significantly influenced by the distance criteria. The second place belongs to RF that selects subset of features and works in splitted feature spaces instead of using a complete feature space. These observations led us to hypothesis that a subset of features (i.e., a specific interval within Ransomware infection period) may improve performance of classification techniques.

5 Proposed method

In the proposed method to overcome high distribution of features, power usage samples are divided into subsamples prior to using different classification techniques to identify the subsamples' labels.

To divide the power usage samples, we assume a fix window size (interval) and move it forward from the starting point of each sample (when the process has actually started), while we append a new subsample to a set of subsamples in each step as depicted in Fig. 6. Algorithm 1 describes the algorithm we used to receive a set of samples and a window size w and generate the subsample's database. Subsample (window) size is a time-value. For example, $w = 6$ means that the subsample contains values for 6 intervals of *PowerTutor*; therefore, its time-length is $6 \times 500_{ms} = 3000_{ms}$.

Input: Sample set $S = \{Sample_1, Sample_2, \ldots, Sample_n\}$ & window size w
Output: Subsamples set DB
$DB \leftarrow \{\}$
for $i \leftarrow 1$ ***to*** n **do**
  $l \leftarrow 1$
  while $(l + w) < length(sample_i)$ **do**
    $Append \quad \langle Label_i, P_l, ., P_{l+w} \rangle \quad to \quad DB$
    $l \leftarrow l + 1$
  end
end
return DB

A label should be assigned to each and every subsample to determine the sample class. As shown in Algorithm 2, a classifier is trained using the subsample database *DB*. The sample is then splitted into a set of subsamples and the Sample Grinding algorithm (1) is used to identify each subsample's label by the trained classifier. This approach identifies the samples class based on the pattern of most similar item in the subsample's database and sets its final label by aggregating all subsamples' labels. Figure 7 illustrates the training phase, and Fig. 8 depicts classification phase of the proposed method.

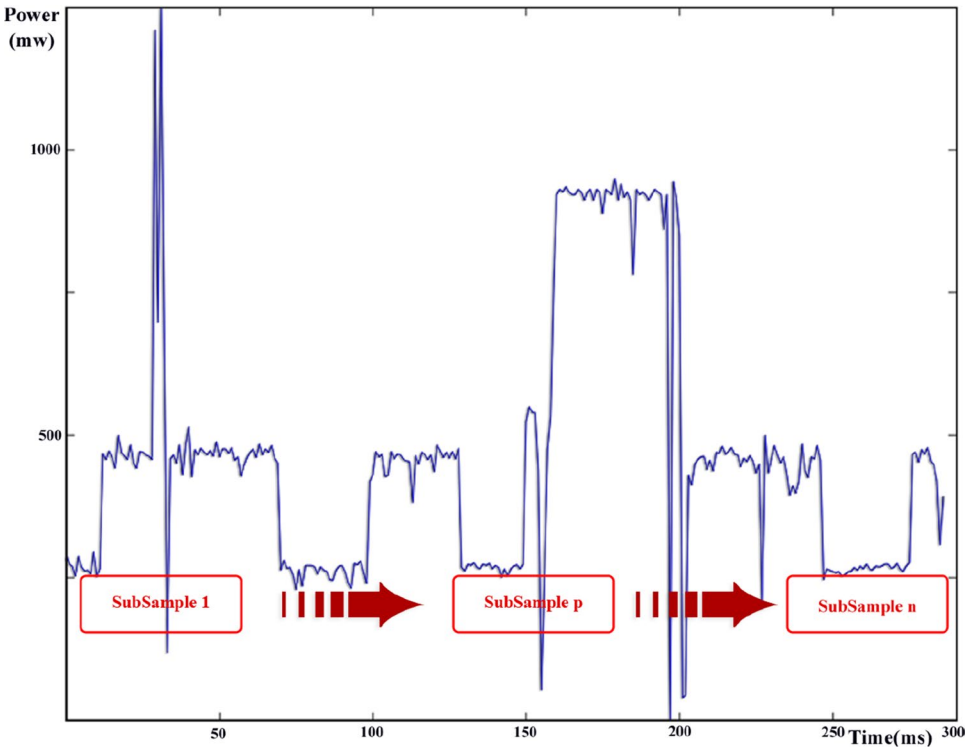

Fig. 6 Subsampling against a sample

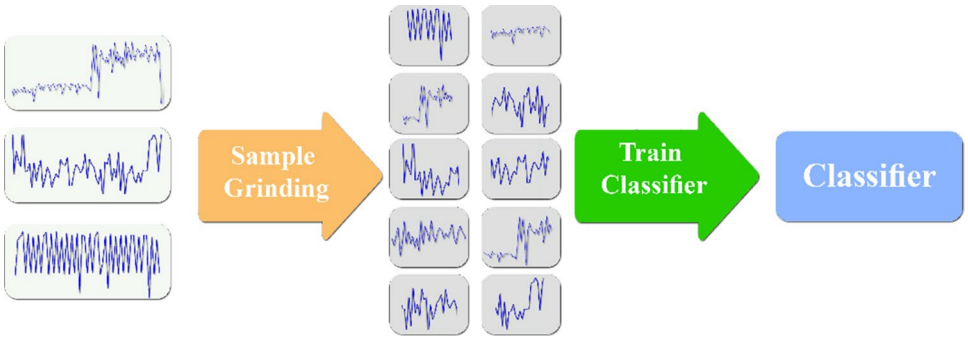

Fig. 7 Training phase of the proposed method

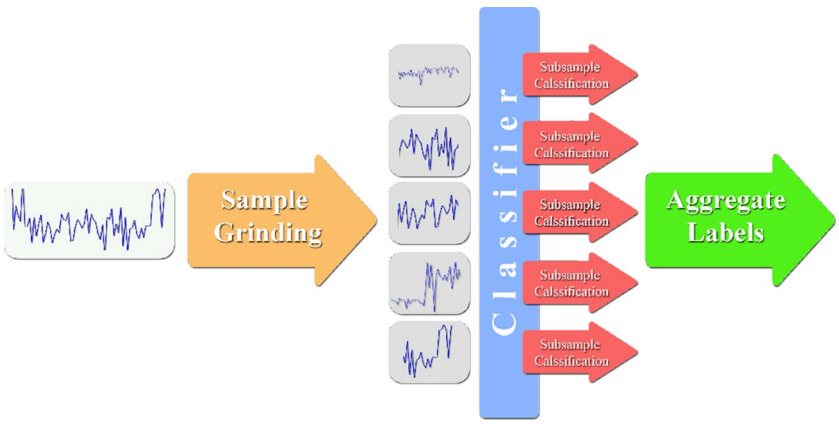

Fig. 8 Classification phase of the proposed method

Table 3 Evaluation metrics for different window sizes, KNN and Euclidean distance: a comparative summary

Window size	Accuracy (%)	Recall (%)	Precision (%)	F-Measure (%)
5	90.67	89.86	84.93	87.32
10	**92.75**	**94.29**	**86.84**	**90.41**
15	92.23	92.86	86.67	89.66
20	91.19	90.00	86.30	88.11
25	86.53	90.00	76.83	82.89
30	87.05	91.43	77.11	83.66
35	86.01	91.43	75.29	82.58
40	81.87	**94.29**	68.04	79.04
45	78.24	91.43	64.00	75.29
50	78.24	91.43	64.00	75.29

Best (optimal) values are highlighted in bold

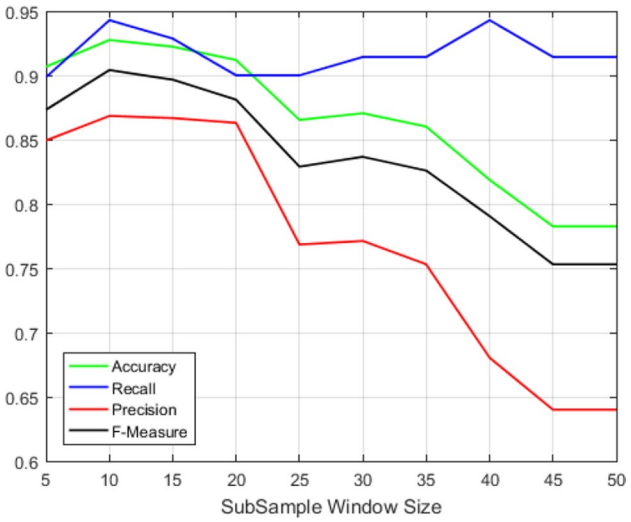

Fig. 9 Evaluation Metrics for different Window Sizes, KNN and Euclidean distance: A Comparative Summary

Input: Subsample database $DB\&Sample$
Output: $Label \epsilon \{R, G\}$
$Labels \leftarrow \{\}$
$Classifier = TrainClassifier(DB)$
$Subsamples = Grind\ Sample\ Using\ Algorithm\ 1$
$i \leftarrow 1$
while $i < size(Subsamples)$ **do**
  $L \leftarrow Classifier(Subsample_i)$
  $Append\ L\ to\ Labels$
  $i \leftarrow i + 1$
end
return $most\ frequent\ item\ in\ Labels$

In the next section, we will discuss our findings.

6 Findings and discussion

We experimented with different window sizes, ranging from 5 to 50 each an increment of 5 in each experiment. Hence, the window size of each dataset includes subsamples with length w.

As discussed in Sect. 5, all subsamples of each sample should be classified. We evaluated grinded data using SVM (see Table 5; Fig. 11), NN (see Table 6; Fig. 12), RF (see Table 7; Fig. 13) and KNN with $K = 1$ classifiers. Table 3 and Fig. 9 show the result for KNN employing the Euclidean distance. Similar setting was applied for KNN using DTW distance and the findings are presented in Table 4 and Fig. 10. In order to summarise the findings and since $K = 1$ is the setting with higher efficiency, other settings ($K = 5, 10$) for KNN are excluded.

Table 4 Evaluation metrics for different window sizes, KNN and DTW distance: a comparative summary

Window size	Accuracy (%)	Recall (%)	Precision (%)	F-Measure (%)
5	89.64	86.96	84.51	85.71
10	91.19	92.75	84.21	88.28
15	**94.27**	**95.65**	**89.19**	**92.31**
20	91.19	94.20	83.33	88.44
25	91.19	94.20	83.33	88.44
30	92.23	92.75	86.49	89.51
35	91.19	91.30	85.14	88.11
40	90.67	89.86	84.93	87.32
45	89.64	86.96	84.51	85.71
50	89.11	84.28	85.51	84.89

Best (optimal) values are highlighted in bold

As shown in Fig. 14, the KNN classifier that uses DTW distance with a subsample size of 7.5 s outperformed all other methods in terms of detection rate 95.65% and performance of 94.27%. Although KNN is the least sophisticated classification approach, it outperformed other rival classification techniques since it only relies on the formation and distribution of goodware's and ransomware's subsamples. The performance of KNN using DTW for all evaluation metrics peaks at *window size* = 15. However, the remaining classifiers were not able to achieve an optimal performance at the specified window size. For example, NN's best accuracy, precision and F-measure occurred at $w = 20$, while highest recall was achieved at $w = 15$. The numerical results indicate that subsamples are not from specified and exact data

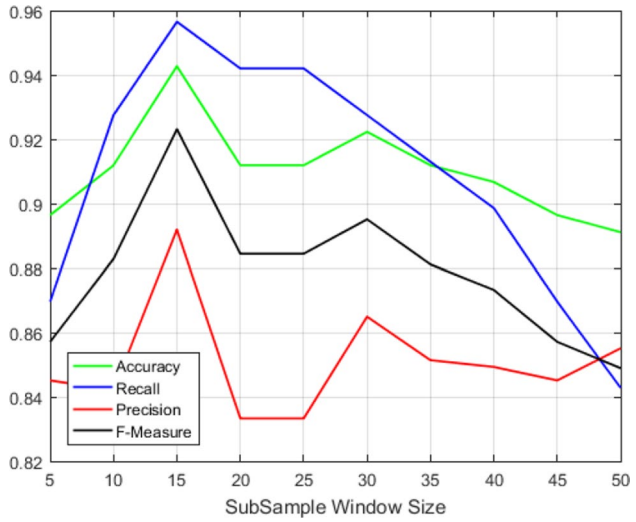

Fig. 10 Evaluation metrics for different window sizes, KNN and DTW distance: a comparative summary

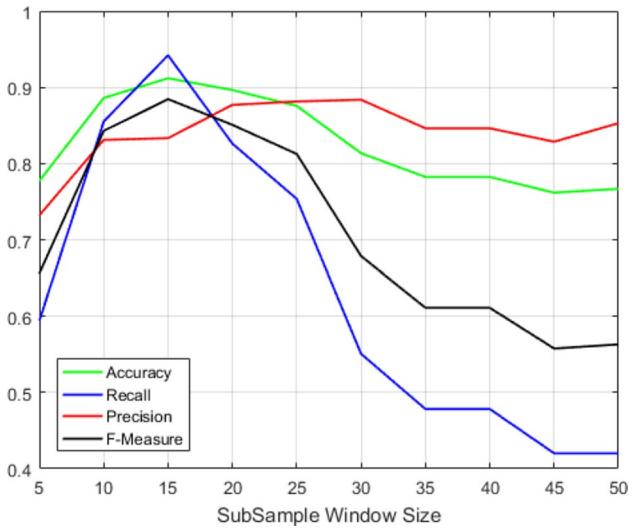

Fig. 11 Evaluation metrics for different window sizes and SVM: a comparative summary

Table 5 Evaluation metrics for different window sizes and SVM: a comparative summary

Window size	Accuracy (%)	Recall (%)	Precision (%)	F-Measure (%)
5	77.72	59.42	73.21	65.60
10	88.60	85.51	83.10	84.29
15	**91.19**	**94.20**	83.33	**88.44**
20	89.64	82.61	87.69	85.07
25	87.56	75.36	88.14	81.25
30	81.35	55.07	**88.37**	67.86
35	78.24	47.83	84.62	61.11
40	78.24	47.83	84.62	61.11
45	76.17	42.03	82.86	55.77
50	76.68	42.03	85.29	56.31

Best (optimal) values are highlighted in bold

Table 6 Evaluation metrics for different window sizes and neural network: a comparative summary

Window size	Accuracy (%)	Recall (%)	Precision (%)	F-Measure (%)
5	88.08	82.61	83.82	83.21
10	88.08	84.06	82.86	83.45
15	89.64	**88.41**	83.56	85.92
20	**90.67**	86.96	**86.96**	**86.96**
25	89.64	85.51	85.51	85.51
30	89.12	85.51	84.29	84.89
35	88.08	82.61	83.82	83.21
40	86.01	81.16	80.00	80.58
45	85.49	82.61	78.08	80.28
50	86.01	82.61%	79.17	80.85

Best (optimal) values are highlighted in bold

distribution and classes have overlap sample(s) in feature space. Therefore, KNN that seeks for most similar subsample to input data outperform other classification approaches. Moreover, according to ability to align subsamples, DTW can find closer energy consumption pattern and consequently provide more accurate classification results than euclidean.

Furthermore and in practice, KNN's requirement for concurrent distance calculations between training and testing objects can be implemented using parallel processing (so distances can be independently computed). Subsamples dictionary can be partitioned into sperate IoT nodes and each subsample is sent to nodes. They return a label and a similarity value and the label having less similarity value is final subsample's label. This approach reduces the classification time and mitigates the need for storage capacity in every node.

7 Conclusion

With increasing prevalence of Internet-connected devices and things in our data-centric society, ensuring the security of IoT networks is vital. Successfully compromised IoT nodes could hold the network to ransom (D'Orazio et al. 2017; Choo 2014). For example, in the case of ransomware, denying availability to data in an IoT network could

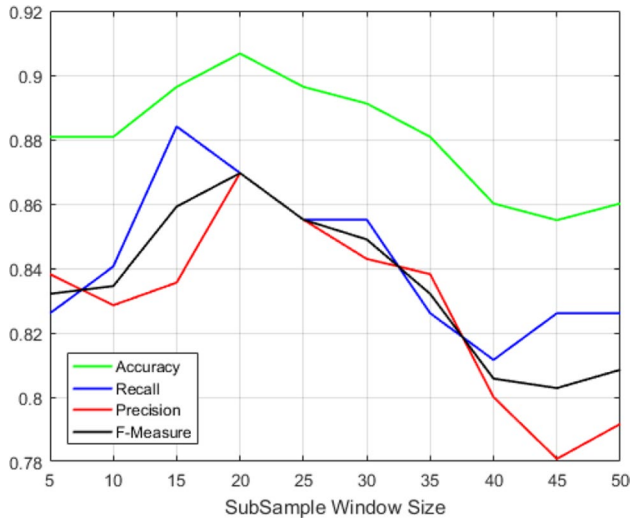

Fig. 12 Evaluation metrics for different window sizes and neural network: a comparative summary

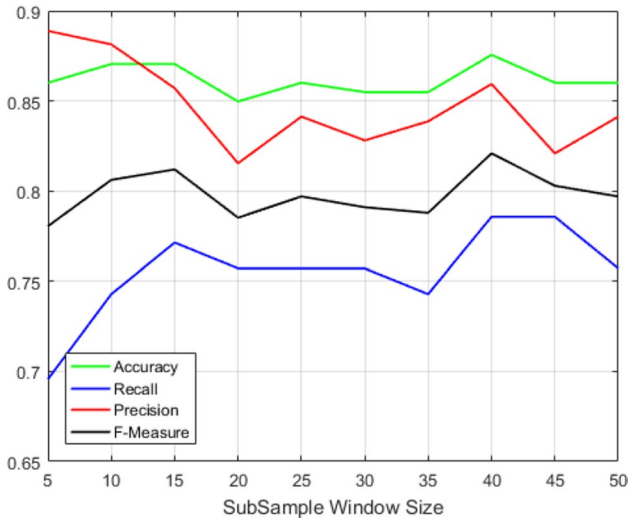

Fig. 13 Evaluation metrics for different window sizes and random forest: a comparative summary

Table 7 Evaluation metrics for different window sizes and random forest: a comparative summary

Window size	Accuracy (%)	Recall (%)	Precision (%)	F-Measure (%)
5	86.01	69.57	**88.89**	78.05
10	87.05	74.29	88.14	80.62
15	87.05	77.14	85.71	**81.20**
20	84.97	75.71	81.54	78.52
25	86.01	75.71	84.13	79.70
30	85.49	75.71	82.81	79.10
35	85.49	74.29	83.87	78.79
40	**87.56**	**78.57**	85.94	82.09
45	86.01	78.57	82.09	80.29
50	86.01	75.71	84.13	79.70

Best (optimal) values are highlighted in bold

adversely affect the operation of an organisation and result in significant financial loss and reputation damage.

In this paper, we presented an approach to detect ransomware, using their power consumption. Specifically, we utilise the unique local fingerprint of ransomware's energy consumption to distinguish ransomware from non-malicious applications. The sequence of applications' energy consumption is splitted into several sequences of power usage subsamples, which are then classified to build aggregated subsample's class labels. Our set of experiments demonstrated that our approach achieved a detection rate of 95.65% and a precision rate of 89.19%.

Future works include prototyping the proposed approach for deploying in a real-world IoT network, with the aims of evaluation and refinement.

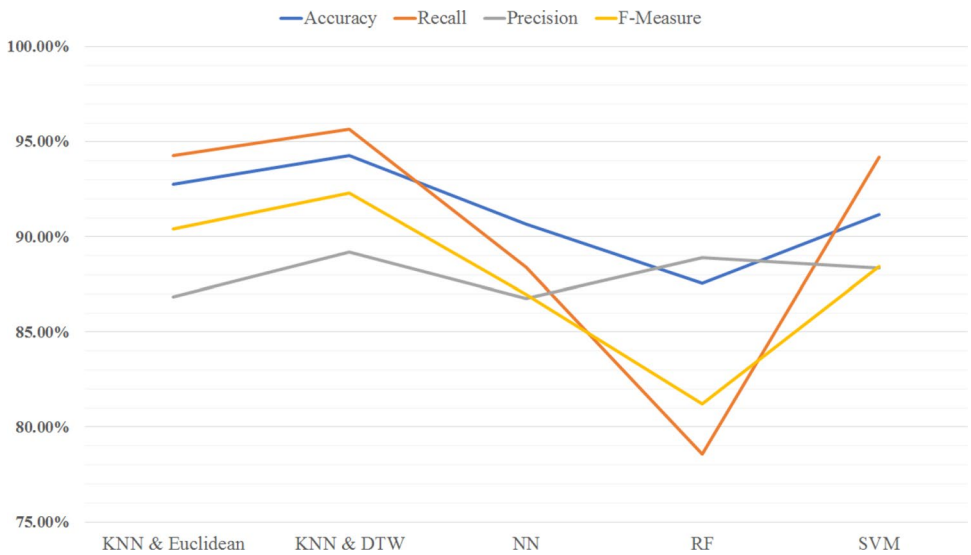

Fig. 14 Best results of each classifier in each measurement: a comparative summary

Acknowledgements We thank VirusTotal for providing us a private API key to access their data for constructing our dataset. This work is partially supported by the European Council International Incoming Fellowship (FP7-PEOPLE-2013-IIF) Grant.

References

Abomhara M, Kien G (2015) Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks. J Cyber Secur 4:65–88

Andronio N, Zanero S, Maggi F (2015) HelDroid: dissecting and detecting mobile Ransomware. In: Proceedings of the 18th international symposium on research in attacks, intrusions, and defenses, Volume 9404. RAID 2015. Springer, New York, pp 382–404

Bertino E, Choo KKR, Georgakopolous D, Nepal S (2016) Internet of things (iot): smart and secure service delivery. ACM Trans Internet Technol 16(4):22:1–22:7

Buczak AL, Guven E (2016) A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun Surv Tutor 18(2):1153–1176

Burges CJ (1998) A tutorial on support vector machines for pattern recognition. Data Min Knowl Disc 2(2):121–167

Caviglione L, Gaggero M, Lalande JF, Mazurczyk W, Urbański M (2016) Seeing the unseen: revealing mobile malware hidden communications via energy consumption and artificial intelligence. IEEE Trans Inf Forensics Secur 11(4):799–810

Choo K-KR (2014) A conceptual interdisciplinary plug-and-play cyber security framework. In: Kaur H, Tao X (eds) ICTs and the millennium development goals: a United Nations perspective. Springer, Boston, pp 81–99

Cover T, Hart P (1967) Nearest neighbor pattern classification. IEEE Trans Inf Theory 13(1):21–27

Damshenas M, Dehghantanha A, Choo K-KR, Mahmud R (2015) M0droid an android behavioral-based malware detection model. J Inf Priv Secur 11(3):141–157

Damshenas M, Dehghantanha A, Mahmoud R (2013) A survey on malware propagation, analysis, and detection. Int J Cyber Secur Digit Forensics 2(4):10–29

Daryabar F, Dehghantanha A, Udzir NI, binti Mohd Sani NF, bin Shamsuddin S (2012) Towards secure model for SCADA systems. In: Proceedings title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), pp 60–64

Dezfouli FN, Dehghantanha A, Eterovic-Soric B, Choo K-KR (2016) Investigating social networking applications on smartphones detecting facebook, twitter, linkedin and google+ artefacts on android and ios platforms. Aust J Forensic Sci 48(4):469–488

D'Orazio CJ, Choo K-KR (2016) Circumventing iOS security mechanisms for APT forensic investigations: a security taxonomy for cloud apps. Future Gener Comput Syst. https://doi.org/10.1016/j.future.2016.11.010

D'Orazio CJ, Choo K-KR, Yang LT (2017) Data exfiltration from internet of things devices: IOS devices as case studies. IEEE Internet Things J 4(2):524–535

Faruki P, Bharmal A, Laxmi V, Ganmoor V, Gaur MS, Conti M, Rajarajan M (2015) Android security: a survey of issues, malware penetration, and defenses. IEEE Commun Surv Tutor 17(2):998–1022

FBI (2016) How to protecting your networks from Ransomware. Tech. rep., USA Government. https://www.justice.gov/criminal-ccips/file/872771/download. Accessed 10 Feb 2017

Fortino G, Trunfio P (2014) Internet of things based on smart objects: technology, middleware and applications. Springer, Berlin

Fu T-C (2011) A review on time series data mining. Eng Appl Artif Intell 24(1):164–181

Gubbi J, Buyya R, Marusic S, Palaniswami M (2013) Internet of things (IOT): a vision, architectural elements, and future directions. Future Gener Comput Syst 29(7):1645–1660

Haykin S (1998) Neural networks: a comprehensive foundation, 2nd edn. Prentice Hall, Upper Saddle River

Jing Q, Vasilakos AV, Wan J, Lu J, Qiu D (2014) Security of the internet of things: perspectives and challenges. Wireless Netw 20(8):2481–2501

Kim H, Smith J, Shin KG (2008) Detecting energy-greedy anomalies and mobile malware variants. In: Proceedings of the 6th international conference on mobile systems, applications, and services. ACM, pp 239–252

Kohavi R et al (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection. Ijcai 14:1137–1145

Kumar JS, Patel DR (2014) A survey on internet of things: security and privacy issues. Int J Comput Appl 90(11):20–26

Mercaldo F, Luo X, Liao Q, Mercaldo F, Nardone V, Santone A, Visaggio CA (2016) Ransomware steals your phone. formal methods rescue it. In: Formal techniques for distributed objects, components, and systems: 36th IFIP WG 6.1 International Conference, FORTE 2016, Held as Part of the 11th International Federated Conference on Distributed Computing Techniques, DisCoTec 2016, Heraklion, Crete, Greece, June 6–9, 2016, Proceedings. Vol. 9688 of Lecture Notes in Computer Science. Springer, pp 212–221

Merlo A, Migliardi M, Caviglione L (2015a) A survey on energy-aware security mechanisms. Pervasive Mob Comput 24:77–90

Merlo A, Migliardi M, Fontanelli P (2015b) Measuring and estimating power consumption in android to support energy-based intrusion detection. J Comput Secur 23(5):611–637

Michalski RS, Carbonell JG, Mitchell TM (2013) Machine learning: an artificial intelligence approach. Artificial intelligence series. Springer, Berlin

Müller M (2007) Dynamic time warping. Springer, Berlin

O'Gorman G, McDonald G (2012) Ransomware: a growing menace. Tech. rep., Symantec Corporation. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/ransomware-a-growing-menace.pdf. Accessed 12 Feb 2017

Pajouh HH, Javidan R, Khayami R, Ali D, Choo K-KR (2016) A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks. IEEE Trans Emerg Top Comput. https://doi.org/10.1109/TETC.2016.2633228

Potlapally NR, Ravi S, Raghunathan A, Jha NK (2006) A study of the energy consumption characteristics of cryptographic algorithms and security protocols. IEEE Trans Mob Comput 5(2):128–143

Sgandurra D, Muñoz-González L, Mohsen R, Lupu EC (2016) Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. arXiv:1609.03020 (preprint)

Shaerpour K, Dehghantanha A, Mahmod R (2013) Trends in android malware detection. J Digit Forensics Secur Law 8(3):21–40

Sicari S, Rizzardi A, Grieco LA, Coen-Porisini A (2015) Security, privacy and trust in internet of things: the road ahead. Comput Netw 76:146–164

Song S, Kim B, Lee S (2016) The effective ransomware prevention technique using process monitoring on android platform. Mob Inf Syst 2016:3–11

Suarez-Tangil G, Tapiador JE, Peris-Lopez P, Ribagorda A (2014) Evolution, detection and analysis of malware for smart devices. IEEE Commun Surv Tutor 16(2):961–987

Tankard C (2015) The security issues of the internet of things. Comput Fraud Secur 2015(9):11–14

Teing Y-Y, Dehghantanha A, Choo K-KR, Yang LT (2017) Forensic investigation of P2P cloud storage services and backbone for IoT networks: BitTorrent Sync as a case study. Comput Electr Eng 58:350–363. https://doi.org/10.1016/j.compeleceng.2016.08.020

Verikas A, Gelzinis A, Bacauskiene M (2011) Mining data with random forests: a survey and results of new tests. Pattern Recogn 44(2):330–349

Watson S, Dehghantanha A (2016) Digital forensics: the missing piece of the internet of things promise. Comput Fraud Secur 2016(6):5–8

Xing Z, Pei J, Keogh E (2010) A brief survey on sequence classification. ACM SIGKDD Explor Newsl 12(1):40–48

Yang H, Tang R (2016) Power consumption based android malware detection. J Electr Comput Eng 2016:1–7

Yang Z (2012) Powertutor—a power monitor for android-based mobile platforms. Tech. rep., EECS, University of Michigan. http://ziyang.eecs.umich.edu/projects/powertutor. Accessed 25 Jan 2017