Received June 7, 2020, accepted July 6, 2020, date of publication July 29, 2020, date of current version August 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3012674

A Pseudo Feedback-Based Annotated TF-IDF Technique for Dynamic Crypto-Ransomware Pre-Encryption Boundary Delineation and Features Extraction

BANDER ALI SALEH AL-RIMY[1], MOHD AIZIANI MAAROF[2], (Member, IEEE), MAMOUN ALAZAB[3], FAWAZ ALSOLAMI[4], SYED ZAINUDEEN MOHD SHAID[2], (Member, IEEE), FUAD A. GHALEB[2], TAWFIK AL-HADHRAMI[5], AND ABDULLAH MARISH ALI[4]

[1]Faculty of Business and Technology, Unitar International University, Petaling Jaya 47301, Malaysia
[2]School of Computing, Universiti Teknologi Malaysia (UTM), Johor Bahru 81310, Malaysia
[3]College of Engineering, IT, and Environment, Charles Darwin University, Casuarina, NT 0810, Australia
[4]Faculty of Computing and Information Technology, King Abdulaziz University (KAU), Jeddah 21589, Saudi Arabia
[5]School of Science and Technology, Nottingham Trent University, Nottingham NG11 8NS, U.K.

Corresponding authors: Bander Ali Saleh Al-Rimy (bnder321@gmail.com), Mamoun Alazab (mamoun.alazab@cdu.edu.au), and Fuad A. Ghaleb (abdulgaleel@utm.my)

This project was funded by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under grant No. (DF-742-611-1441). The authors, therefore, gratefully acknowledge the DSR technical and financial support.

ABSTRACT The cryptography employed against user files makes the effect of crypto-ransomware attacks irreversible even after detection and removal. Thus, detecting such attacks early, i.e. during pre-encryption phase before the encryption takes place is necessary. Existing crypto-ransomware early detection solutions use a fixed time-based thresholding approach to determine the pre-encryption phase boundaries. However, the fixed time thresholding approach implies that all samples start the encryption at the same time. Such assumption does not necessarily hold for all samples as the time for the main sabotage to start varies among different crypto-ransomware families due to the obfuscation techniques employed by the malware to change its attack strategies and evade detection, which generates different attack behaviors. Additionally, the lack of sufficient data at the early phases of the attack adversely affects the ability of feature extraction techniques in early detection models to perceive the characteristics of the attacks, which, consequently, decreases the detection accuracy. Therefore, this paper proposes a Dynamic Pre-encryption Boundary Delineation and Feature Extraction (DPBD-FE) scheme that determines the boundary of the pre-encryption phase, from which the features are extracted and selected more accurately. Unlike the fixed thresholding employed by the extant works, DPBD-FE tracks the pre-encryption phase for each instance individually based on the first occurrence of any cryptography-related APIs. Then, an annotated Term Frequency-Inverse Document Frequency (aTF-IDF) technique was utilized to extract the features from runtime data generated during the pre-encryption phase of crypto-ransomware attacks. The aTF-IDF overcomes the challenge of insufficient attack patterns during the early phases of the attack lifecycle. The experimental evaluation shows that DPBD-FE was able to determine the pre-encryption boundaries and extract the features related to this phase more accurately compared to related works.

INDEX TERMS Cybersecurity, early detection, malware, ransomware, TF-IDF.

I. INTRODUCTION

Crypto-ransomware is a malware category that employs the encryption against personal files and business data in victim's machine and deny the access to these resources. The irreversible effect is what characterizes crypto-ransomware attacks from malware attacks [1]. Therefore, it is imperative to detect these attacks early at the pre-encryption phase, during which the crypto-ransomware installs itself in victim's machine and conducts reconnaissance to

The associate editor coordinating the review of this manuscript and approving it for publication was Constantinos Marios Angelopoulos.

discover the running environment and locate the targeted files. Therefore, the effectiveness of any early detection solution is governed by the accurate delineation of the pre-encryption boundaries. Extant research [2]–[5] have applied a fixed time-based thresholding approach, to determine the pre-encryption phase boundaries, from which the attack patterns have been acquired and used to build the detection models. However, the fixed time thresholding approach implies that all instances start the encryption at the same time. Such assumption does not necessarily hold for all samples as the time for the main sabotage to start varies among different crypto-ransomware instances due to the obfuscation techniques employed to change the attack pattern and evade detection, which generates different attack behavior for each sample or family [6]–[8]. Therefore, fixed time thresholding approach could miss the beginning of encryption process and; consequently; the captured data would not accurately represent the pre-encryption phase of crypto-ransomware attacks.

Moreover, the insufficient behavioral data acquired during the early phases of crypto-ransomware attacks adversely affects the ability of features extraction techniques to weighing the features accurately. During features extraction, the general-purpose APIs could be given weights higher than what they should get in reality. Consequently, the extracted features might not correctly represent the pre-encryption phase of crypto-ransomware attacks. Although Term Frequency-Inverse Document Frequency (TF-IDF) can calculate features values more accurately than other techniques like TF [7], the issue with applying it on the small portion of the data raises when calculating the IDF term. That is, a particular API might have a small Document Frequency (DF) value when only considering the pre-encryption data while; in reality, holds high DF value when considering the entire attack's data. As such, the TF-IDF value of that API will be high on pre-encryption data but low on the entire data. Consequently, TF-IDF will erroneously give higher weight to that API when relying only on the pre-encryption data while; in reality; it is a general-purpose API with respect to the entire data and should be penalized instead. Such calculation obstructs the ability of TF-IDF to give accurate numerical representation to the extracted features.

To this end, this paper proposes a Dynamic Pre-encryption Boundary Delineation and Feature Extraction (DPBD-FE) scheme that dynamically determines the boundaries of the pre-encryption phase in crypto-ransomware lifecycle. Contrary to the fixed thresholding employed by the extant works, the DPBD-FE tracks the pre-encryption phase for each instance individually based on the first occurrence of any cryptography-related APIs. The intuition is that, the crypto-ransomware needs to call a cryptography-related API and/or function to start the real sabotage [9], [10]. Therefore, the first call of any of those cryptography-related APIs could be an indicator of an imminent encryption process. Consequently, the first cryptography-related API is located in the boarders between pre-encryption phase and the encryption phase. As such, the pre-encryption data could be easily and accurately identified and acquired regardless of the time each sample takes before it starts the encryption. Based on such data, pre-encryption attack patterns and features can be extracted and used to train more effective and accurate early detection models. To extract the pre-encryption features, the annotated Term Frequency-Inverse Document Frequency (aTF-IDF) technique was proposed.

Unlike traditional TF-IDF, the proposed aTF-IDF is aware of the unobserved behavioral patterns that come after the pre-encryption boundary. As such, it overcomes the issue of insufficient behavioral patterns. With that, the proposed technique becomes able to distinguish and penalize the general-purpose APIs. Those APIs are normally utilized by all programs, whether benign or malicious, and add no information about a particular set of programs. As such, aTF-IDF avoids the issue of highlighting the APIs that seem to be ransomware-related when looking to the pre-encryption data only, while they are general purpose and useless given the full-length dataset. The contribution of this paper is four-fold.

a) A Dynamic Pre-encryption Boundary Delineation (DPBD) techniques was proposed to dynamically determine the boundaries of pre-encryption phase in crypto-ransomware lifecycle.

b) An annotated TF-IDF (aTF-IDF) techniques was proposed to overcome data insufficiency when extracting the attack features at the pre-encryption phase.

c) Both DPBD and aTF-IDF were incorporated to the proposed DPBD-FE scheme for accurate pre-encryption boundary definition and feature extraction in order to increase the efficacy and accuracy of the crypto-ransomware early detection model.

d) An extensive experimental evaluation was conducted to show the improvement that DPBD-EF scheme had achieved.

For the purpose of this study, crypto-ransomware and ransomware are used interchangeably unless stated otherwise. The rest of this paper is organized as follows. Section II gives an overview about the related work. Section III elaborates on the crypto-ransomware attack model. The methodology followed to design and develop the proposed techniques is discussed in Section VI. In Section V, the experimental results, were explained while Section VI analyses and discuss those evaluation results with the comparison with the related works. The paper is concluded with Section VII by a summary of the methods and results as well as suggestions for future work.

II. RELATED WORKS

Ransomware is a malware category that locks user data and files and demands ransom to release them [11]–[14]. Since its emergence on the late 1980s, ransomware becomes a major threat that intimidate the accessibility to both personal and business data [15]. As its name implies, the main purpose of ransomware is to blackmail victims by holding their digital assets to ransom [16], [17]. Attackers have developed many variants of ransomware to achieve such a goal, which explains the dominance of ransomware attacks in the threat landscape recently [4], [18]–[20].

In 2014, the total loss due to ransomware attacks reached $3 million [4]. Moreover, it was reported that in 2015 the attackers earned around $352 million from victims around the world [21]. In Indiana county only, victims paid around $220K in 2016 to recover from ransomware attacks [21].

Ransomware is categorized into two types, locking-ransomware and crypto-ransomware [1]. Unlike locking-ransomware whose damage could be easily circumvented, the effect of crypto-ransomware is not reversible as it employs the encryption against user files. Consequently, without the decryption key, it becomes difficult, if not impossible, for the victim to access his data again [22]. As such, it is imperative to detect this type of malware as early as before it starts encrypting user files and data.

Several studies have been devoted to address this problem and proposed solutions to detect ransomware attacks. These solutions can be categorized into two types, data-centric, and process-centric. Data-centric ransomware detection monitors the digital assets in victim's machine and raises an alarm when suspicious change is detected [17]. The decoy, entropy, and similarity techniques were employed by these data-centric solutions to monitor the file structure before and after it got accessed [8], [15], [23]–[25]. However, this approach is unable to differentiate between the changes caused by the crypto-ransomware form those caused by the benign programs, which lead to high rate of false alarms [9], [26], [27]. More importantly, this approach does not fully protect from ransomware attacks as it sacrifices part of the files that might be more valuable than the remaining data [26], [28]. Consequently, the data-centric approach is not effective for the early detection of crypto-ransomware.

In the process-centric studies, on the other hand, monitoring the behavior of the running process is used to collect different types of behavioral data, which were then used for training different machine learning classifiers like Random Forests and Naïve Bayes [21], [24], [29], [30]. However, these solutions rely on the entire runtime data; which include pre-encryption and post-encryption data; to train the detection model [3], [31]. This approach is based on the premise that entire attack patterns are fully available at the detection time, which does not hold for crypto-ransomware early detection where the data about the attack are not fully available [3]. Monitoring the different resources in the local machine, like CPU, network, I/O buffer and memory is another type of process-centric ransomware detection [8], [18], [23], [32], [33]. When some events related to encryption were observed, the detection system raises an alarm. However, the reliance on ad-hoc events to detect crypto-ransomware attack increases the rate of false alarms as those events are not mutually exclusive to crypto-ransomware. Some normal programs also raise similar events [9]. Furthermore, those ad-hoc events could be raised after the encryption, which makes this approach ineffective for the early detection [8]. As such, it is necessary for an effective detection to take place during the early phases (pre-encryption) before the ransomware starts its main sabotage, i.e. the encryption.

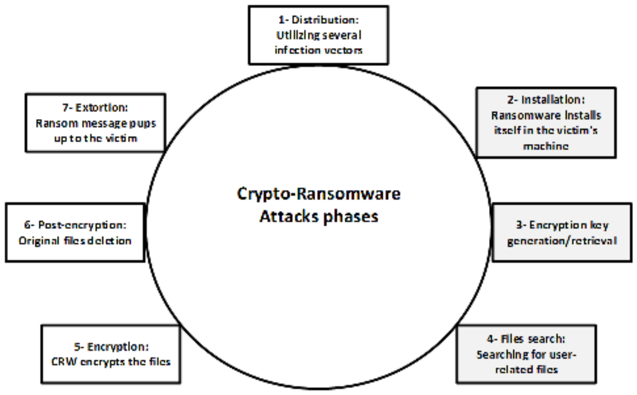

1- Distribution: Utilizing several infection vectors
2- Installation: Ransomware installs itself in the victim's machine
3- Encryption key generation/retrieval
4- Files search: Searching for user-related files
5- Encryption: CRW encrypts the files
6- Post-encryption: Original files deletion
7- Extortion: Ransom message pups up to the victim
Crypto-Ransomware Attacks phases

FIGURE 1. **Ransomware attack model.**

III. ATTACK MODEL

Ransomware's lifecycle starts from the moment when the malicious code is disseminated and lasts until the financial claim is shown to the victim. During this lifecycle several actions are conducted in order to successfully hijack the user's files and resources. According to [8], [18], [21], [34]–[39], ransomware attacks go through several essential phases as illustrated in Figure 1 and summarized below.

a) DISTRIBUTION: During this phase, the ransomware is packed and delivered into the victim's system using different exploitation techniques such as email attachment or drive-by download.

b) INSTALLATION: In this phase, crypto-ransomware installs itself in the victim's machine. Such installation also involves exploring the running environment and collecting information about the victim's device, such as platform type, OS version, and already-installed programs.

c) ENCRYPTION KEY GENERATION: Crypto-ransomware retrieves the encryption key from the C&C server or generates it locally.

d) FILES SEARCH: Ransomware starts looking for targeted files.

e) ENCRYPTION: Based on the attack approach, Crypto-ransomware starts encrypting the targeted files either one by one concurrently with the files search process or waits until listing all the files then encrypting them all at once.

f) POST ENCRYPTION: Once the encryption process is finished, the original files are either deleted or moved to another location with new names.

g) EXTORTION: After encrypting and deleting/moving all files, the extortion message is shown to the victim asking for a ransom accompanied by payment instructions.

Among the stages enlisted above, the pre-encryption phase of crypto-ransomware attacks lifecycle involves the (b) installation, (c) encryption key generation and (d) file search.

IV. METHODOLOGY

In this section, we describe the design of proposed Dynamic Pre-encryption Boundary Delineation and Feature Extraction scheme (DPBD-FE) technique that accurately determines the pre-encryption boundaries of crypto-ransomware lifecycle and extract the discriminative feature that represent the attack patterns during this phase. We start by elaborating about the design and implementation of the proposed DPBD for the pre-encryption boundary delineation. Likewise, the design and implementation of the proposed aTF-IDF technique for pre-encryption features extraction is detailed. In this study, the dynamic analysis was employed as it is the effective approach that overcome the polymorphic and packing techniques that crypto-ransomware attacks use to resist and evade detection. In addition, the early detection evokes the dynamic analysis as it intends to detect the attacks during the early phases of the runtime.

A. ATTACK PATTERNS ACQUISITION

To collect the attack's patterns from runtime data of the samples in the dataset, each sample was submitted to the analyzing machine and underwent the dynamic analysis as elaborated in sub section A under Section II. Once the instance was submitted, the sandbox agent in guest machine hooks the process created by that sample and captures the runtime data including the API calls and dumps them into a respective trace file specified for that sample. Those files constitute the corpus, from which the dataset was built, and the features were extracted and selected before they were used to train the detection model. Figure 2 shows the architecture of crypto-ransomware dynamic analysis. After each run, the guest machine was reverted to its original, clean state to ensure that the next sample will not be affected by the previous infection. From the runtime data in each trace file, only the API calls along with the parameters were kept and all other data were discarded. These APIs constitute the input for the DPBD-FE scheme as will be elaborated in the subsequent sections.

B. DYNAMIC PRE-ENCRYPTION BOUNDARY DELINEATION TECHNIQUE FOR PRE-ENCRYPTION DATA EXTRACTION

To accurately delineate the boundary of the pre-encryption phase, the vector space model along with Rocchio relevance feedback were utilized to design and implement the Dynamic Pre-encryption Boundary Delineation (DPBD) technique, by which the pre-encryption boundary vector was built. This vector contains all cryptography related APIs that crypto-ransomware instances have called during the attacks. The vector constitutes the boundary of pre-encryption phase of the crypto-ransomware attacks such that the first occurrence of any of the vector's entries represents the boundary between the end of pre-encryption phase and the

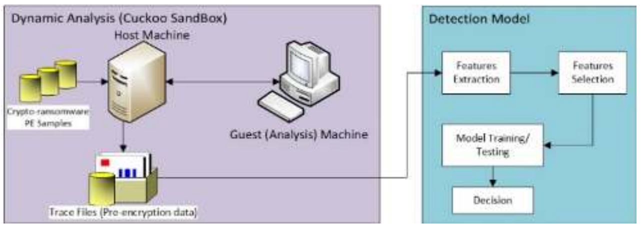


FIGURE 2. The dynamic crypto-ransomware analysis and detection process.

CryptDecodeObjectex	CryptDecrypt	CryptUnprotectData
CryptGetObjectUrl	CryptReleaseContext	CryptCreateHash
CryptHashData	CryptHashData	CryptGetHashParam
CryptExportKey	CryptGenKey	CryptCreateHash
EncryptMessage	CryptAcquireContexta	CertGetNameStringW

FIGURE 3. Examples of cryptoAPIs.

beginning of the encryption phase of the attacks. The intuition is that, the calling of any of cryptography-related APIs indicates an imminent encryption process which could be the beginning of actual attack against user files and data. Unlike the fixed time-based thresholding employed by existing works, DPBD tracks the encryption starting point for each instance and accurately determines the boundary of the pre-encryption phase based on the cryptography-related APIs that crypto-ransomware instance used to encrypt user files. To build the boundary vector, DPBD started by defining an initial boundary vector, by which the data of each instance were divided into three subsets, namely initPre, initAt and initPost. Based on those subsets, the final boundary vector was built using the Pseudo Rocchio Relevance Feedback.

1) BUILDING THE INITIAL SUBSETS

At the outset, an exploration process was carried out to build an initial boundary vector called V_{init} out of the explicit CryptoAPI calls, i.e. the cryptography-specific APIs, according to [40]. These cryptoAPIs were used as seeds for the pre-encryption boundary vector. Figure 3 shows a snippet of these cryptoAPIs. In this process, the explicit CryptoAPI calls were determined and gathered into V_{init}. Then, V_{init} was utilized to divide the raw data in each trace file into three parts, initPre, initAt, and initPost. The initPre is composed of the API entries starting from the first API in the trace file until the first occurrence of any entry of V_{init}. Similarly, initAt is composed of the APIs that are contained between the first and last occurrence of any entry of V_{init}. Likewise, the initPost contains the APIs that are collected right after the last occurrence of any V_{init} entries until the end of the trace file. The purpose of such division is to extract the APIs that were called during the encryption phase, i.e. the API calls contained within initAt, and determine the API calls associated with the encryption process. By adding these APIs into the boundary vector, the technique becomes able to determine the pre-encryption boundary even before the attack starts using the explicit CryptoAPIs and/or encryption functions.

2) PSEUDO ROCCHIO RELEVANCE FEEDBACK

As mentioned previously, pre-encryption boundary is the point whereby crypto-ransomware starts using cryptography-related APIs and/or functions. Cryptography-related APIs include the explicit cryptoAPIs as well as other APIs that use any of explicit cryptoAPIs and/or functions as a parameter. Those APIs are contained within the initAt subset. Thus, the initAt subset is the area of interest whereby the entries of the pre-encryption boundary vector can be found. However, not all APIs in initAt are cryptography-related, so it is not suitable to include all of them in the final boundary vector as they could be found at the very beginning of the trace file which leads to exclude many pre-encryption data and deprives the model from useful pre-encryption patterns. To filter out the unrelated APIs, Pseudo Rocchio Relevance Feedback (PRRF) technique was proposed to calculate the weight for each API in initAt subset and keep only the ones with weights higher than the threshold. The original Rocchio relevance feedback calculates the relevancy of query terms according to Eq. 1.

$$V_f = \frac{1}{n}\sum_{relevant} d_j - \frac{1}{N-n}\sum_{irrelevant} d_j \tag{1}$$

where V_f denotes the feedback vector; N denotes the corpus size; n denotes number of relevant trace files; $\sum_{relevant} d_j$ is the relevant set and $\sum_{irrelevant} d_j$ is the irrelevant set.

The PRRF technique utilized the Term Frequency-Inverse Document Frequency (TF-IDF) to build two vectors, namely relevant and irrelevant vectors. The relevant vector was built by applying TF-IDF on the initAt subset. This vector is called relevant because it was built based on the data contained within initAt which consists of all APIs that were collected during the encryption period, i.e. between the first and last occurrence of the cryptoAPIs in V_{init}. Similarly, the irrelevant vector was built by applying TF-IDF on both initPre and initPost together. Once the vectors were built, the enhanced Rocchio technique proposed by Salton and Buckley [41] was applied as shown by Eq. 2.

$$V_f = V_{init} + \frac{1}{n_1}\sum_{relevant} d_j - \frac{1}{n_2}\sum_{irrelevant} d_j \tag{2}$$

where V_f denotes the feedback vector; V_{init} is the original (initial) vector; $\sum_{relevant} d_j$ is the relevant set (the iniAt in our case) and $\sum_{irrelevant} d_j$ is the irrelevant set (initPre and initPost). As irrelevant set was composed of initPre and initPost while the relevant subset was composed of the initAt subset of the same crypto-ransomware instances, $n_2 = 2 \times n_1$. Therefore, equation (2), can be rewritten as in Eq. 3.

$$V_f = V_{init} + \frac{1}{n}\Big(\sum_{relevant} d_j - \frac{1}{2}\sum_{irrelevant} d_j\Big) \tag{3}$$

The pre-encryption boundary vector was derived from V_f based on a predefined threshold such that, the APIs whose TF-IDF values were greater than or equal to the threshold were included in the selected vector V_s. This threshold was determined by calculating the average of TF-IDF values of V_{init} entries. To preserve all cryptoAPIs in the pre-encryption boundary vector, V_{init} and V_s were fused into V_b [41]. Therefore, the vector V_b is the pre-encryption boundary vector. Figure 4 shows the design of DPBD technique while Figure 5 shows its pseudo code.

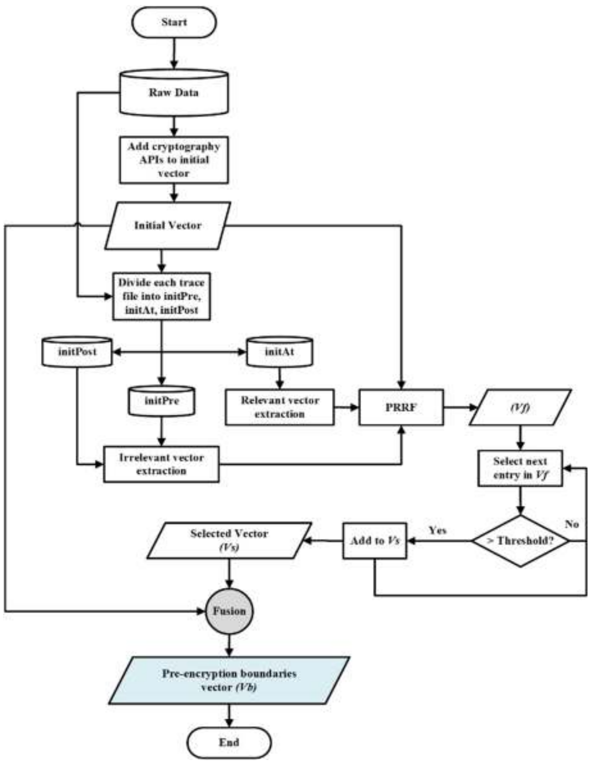

Start
Raw Data
Add cryptography APIs to initial vector
Initial Vector
Divide each trace file into initPre, initAt, initPost
initPost
initAt
initPre
Relevant vector extraction
PRRF
(Vf)
Irrelevant vector extraction
Select next entry in Vf
Selected Vector (Vs)
Add to Vs
Yes
> Threshold?
No
Fusion
Pre-encryption boundaries vector (Vb)
End

FIGURE 4. The Dynamic Pre-encryption Boundary Definition (DPBD) technique.

C. ANNOTATED TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY TECHNIQUE FOR PRE-ENCRYPTION FEATURES EXTRACTION

In this section, the annotated Term Frequency-Inverse Document Frequency (aTF-IDF) technique is proposed to extract the features related to pre-encryption phase of the crypto-ransomware lifecycle. The problem with applying the traditional TF-IDF on pre-encryption data raises when calculating the IDF term. That is, a particular API might have a small Document Frequency (DF) value when only considering the pre-encryption data but high DF value when considering the entire data. In this case, the TF-IDF value of that API will be high only on pre-encryption data while low on the entire data. Consequently, the API will be given a more importance (weight) when relying only on the pre-encryption data which indicates that it is an attack-specific API while; in reality; it is a general-purpose API with respect to the entire data and should be given lower TF-IDF value instead. Such general-purpose APIs have low prediction meaning and might be utilized by the malicious program for obfuscation purposes. As such, they should be penalized [42]. The aTF-IDF addresses this issue by employing the annotation to highlight the APIs that are more related to pre-encryption phase without counting the general-purpose APIs.

The general formula used to calculate Term Frequency-Invers Document Frequency (TF-IDF) is shown in Eq. 4

$$w\left(api_k^j\right) = tf\left(api_k^j\right) \cdot \log \frac{N}{idf\,(api_k)} \tag{4}$$

where api_k denotes the k$^{\text{th}}$ API; $tf\,(api_k^j)$ is the term frequency that calculates how many times the api_k was called by the ransomware instance r_j in the subset. Similarly, $idf\,(api_k)$ is the inverse document frequency that calculates how many ransomware instances r_j in the subset called api_k at least ones, while N denotes the total number of ransomware instances in the subset. Before calculating $w(api_k^j)$, each $tf\,(api_k^j)$ was normalized to prevent TF-IDF from favouring the non-informative APIs in long trace files over the informative ones in shorter trace files. The normalization was carried out according to Eq. 5 by dividing the $tf\left(api_k^j\right)$ of each instance by $length(tr_j)$; where tr_j denotes the trace file of crypto-ransomware instance r_j. As such, the normalized TF-IDF was calculated according to Eq. 6.

$$tf'\left(x_k^j\right) = \frac{tf\,(x_k^j)}{length(tr_j)} \tag{5}$$

$$w'\left(x_k^j\right) = tf'\left(x_k^j\right) \cdot log \frac{N}{idf\ (x_k)} \tag{6}$$

Unlike traditional TF-IDF used by extant crypto-ransomware like [2], [30], [42]–[46], the annotated TF-IDF (aTF-IDF) distinguishes the APIs that are called during the pre-encryption phase from those who are called during and after the encryption. Therefore, aTF-IDF can distinguish whether a particular API is a general-purpose even if it got high DF weight in the pre-encryption data. The aTF-IDF started by labelling (annotating) each API in the original dataset according to the location in the trace file from the pre-encryption boundary V_b. That is, with the help of the boundary vector V_b, each API was observed against V_b to see whether it is located before or after the boundary. Namely, APIs in the original dataset were observed one by one and annotated as ‘pre’ until encountering any of V_b entries. After encountering the V_b entry, the annotation is changed into ‘enc’ for all subsequent APIs. Given the annotated APIs in each trace file, the features vector for the ‘pre’ subset was built and the aTF-IDF weight was calculated according to Eq. 7. The aTF-IDF is composed of two terms, namely aTF and IDF. The aTF term is calculated according to Eq. 8.

$$w\left(api_k^j\right) = atf\left(api_k^j\right) \cdot \log \frac{N}{idf\,(api_k)} \tag{7}$$

$$atf\left(api_k^j\right) = C\left(\varphi \times api_k^j\right) \tag{8}$$

where $C(\varphi \times api_k^j)$ is the term frequency of the api_k in crypto-ransomware instance r_j; $\varphi = \{0, 1\}$ is determined based on the api_k^j's annotation, according to Eq. 9.

$$\varphi = \begin{cases} 1 & if\ annotation\ is\ pre \\ 0 & otherwise \end{cases} \tag{9}$$

As shown by Eq. 6, aTF-IDF, penalizes the common APIs that are called by most or all instances. These APIs are considered general-purpose APIs that do not add information about the target type. Therefore, the denominator of the equation increases according to the number of instances that contain that feature api_k which decrease the value of $w(api_j^k)$. The pre-encryption aTF-IDF weights were stored in a vector that represents the pre-encryption phase of ransomware lifecycle. Figure 5 shows the pseudo code of the feature’s extraction using aTF-IDF technique.

V. THE EXPERIMENTAL RESULTS

This section describes the setup of experimental environment in which, the implementation of the proposed scheme and techniques was conducted. Then the dataset used by this study was detailed explaining the different instances and the type of data that have been acquired. The experimental results of each technique were presented including the comparison with the previous studies.

A. THE EXPERIMENTAL ENVIRONMENT SETUP

The experiments were conducted on a controlled environment built on a machine with Intel(R) Core (TM) i7-4790 CPU @ 3.60 GHZ and 16 GB RAM. The analysis environment was built according to [47]. The Cuckoo Sandbox; a well-known and widely used malware analysis platform; was used as analysis environment [43], [48], [49]. The VMware Technology was utilized to build the sandbox. Within this sandbox, the host machine was created using Linux Ubuntu 4.4.0-59-generic. Then, the VirtualBox was utilized to create the Windows 32-bit based guest machine. To create a realistic testing environment, several programs and applications like MS Office, Adobe Acrobat Reader, Google Chrome and Mozilla Firefox were installed in the guest machine. In addition, user-alike folders were created in different locations in the local storage of the guest machine. Around 925 files, including MS Word documents, Excel, PPT, Visio, PDF, JPG and short video files were stored in these folders. These files were attacked during the dynamic analysis when crypto-ransomware samples were executed within the Sandbox environment. Those files include, but not limited to word documents, excel sheets, power point, pdf, jpeg and gif files. Crypto-ransomware and benign programs were run one by one. For each program, the data were dumped into an independent trace file. Those trace files contain the API calls used by the program under analysis during the runtime. After each run, the guest machine was restored into the original, clean state. Extracted data was gathered and the features were extracted and selected during the pre-processing phase. Once ready, the dataset was used to build the detection model. The proposed techniques as well as results and analysis were

TABLE 1. Crypto-ransomware families used by this study technique.

Family	Year	Techniques	Target	# samples
Cryptolocker	2013	RSA	User files	6,450
Cryptowall	2014	RSA 2048-bit	User files	3,309
Cryrar	2012	RAR-sfx	User files	1,743
Locky	2016	RSA-2048 + AES-128 cipher with ECB mode	User files	2,661
Petya	2016	AES-128	MBR, user files	2,846
Reventon	2012	N/A	user files	2,798
Teslactypt	2015	ECC	Games and multimedia files	1,670
Wannacry	2017	RSA 2048 bit	User files	1,899
Cerber	2016	RSA-2048	User files	1,203
Filecryptor	2013	2048-bit RSA	User files	3,428
Crypt	N/A	N/A	User files	3,672
CTB_Locker	2014	ECC	User files	2,701
Satana	2016	256-bit AES in ECB	User files	1,258
CryptXXX	2016	RSA4096	User files	2,934
Sage	2016	AES 256 and RSA 1024	User files	806

implemented using Python libraries including Sklearn, Pandas and Numpy.

B. THE DATASET

Table 1 shows the list of crypto-ransomware families used by this study. The corpus of crypto-ransomware binaries used in this study were downloaded from http://www.virusshare.com public repository [5], [29], [50], [51]. The corpus consists of 39,378 samples. Those samples represent different fam ilies such as Cerber, TeslaCrypt, CryptoWall, Petya and WannaCry.. In addition, 16057 benign programs were downloaded from informer.com, the well-known Windows software repository. informer.com [5], [29], [52], [53], a popular Windows-based applications repository. Then, both ransomware and benign programs were run in the sandbox and dynamically analyzed [54], [55].

C. EXPERIMENTAL RESULTS

The pre-encryption boundary vector is shown in Table 2. Each API represents an entry in the boundary vector. The weight of each entry is shown also in the table as calculated by the equation (3). The average of API weights was calculated according to Eq. 10 to determine the threshold by which, the APIs whose weights are higher than the threshold were included into the vector.

$$W_{avg} = \frac{1}{M} \sum_{i=0}^{i=m-1} w_i \tag{10}$$

where w_{avg} denotes the average of vector's weights; w_i denotes the weight of the ith API; and $M = \{0, 1, 2, \ldots, m-1, m$ is the entire API calls.

To measure the accuracy of the DPBD technique in specifying the pre-encryption boundary, event-based procedure proposed by [4] was employed to monitor cryptography-related events (calls) raised by crypto-ransomware samples during the attack. The fraction of crypto-ransomware samples that trigger the cryptography-related events during the pre-encryption phase has been

TABLE 2. The pre-encryption boundary vector.

API	Weight
Cryptdecrypt	0.806
Cryptencrypt	0.621
Certcontrolstore	0.612
cryptacquirecontexta	0.479
crypthashdata	0.423
cryptexportkey	0.404
cryptcreatehash	0.393
certopenstore	0.381
cryptdecodeobjectex	0.356
ntdeletefile	0.324
cryptunprotectdata	0.321
ntdeletevaluekey	0.309
decryptmessage	0.301
encryptmessage	0.301
cryptgenkey	0.279
rtlcompressbuffer	0.247
internetgetconnectedstate	0.243
cryptacquirecontextw	0.203
ntcreateprocessex	0.197
removedirectorya	0.190
ntqueryfullattributesfile	0.189
openservicea	0.158

determined. The fraction of crypto-ransomware instances, R_d, that starts any cryptography-related activity before the proposed DPBD model detects them was determined according to equation (11).

$$R_d = 1 - \frac{\beta}{N} \sum_{i=1}^{i=N} r_i \tag{11}$$

where r_i denotes the i^{th} crypto-ransomware sample, N is the total number of crypto-ransomware samples in the dataset, β denotes whether the sample was detected before the ransomware called any of cryptography-related API for the first time and is determined according to Eq. 12. The same procedure was also applied on the datasets built using the time-based thresholding employed by existing works [3]–[5].

$$\beta = \begin{cases} 1 & \textit{if detected before calling the 1st crypto API} \\ 0 & \textit{otherwise} \end{cases} \tag{12}$$

Pseudo code 1: DPBD Technique

Input: CRW PE Corpus

Output: Pre-encryption boundaries vector V_b.

```
1:  Run PE files
2:  Collect APIs into trace file
3:  Retrieve all $CryptoAPI_{exp}$
4:  $V_{init} \leftarrow CryptoAPI_{exp}$
5:  For each trace file in 2:
6:      while $V_{init}$ not encountered:
7:          $pre \leftarrow API$
8:      end, go to 7
9:      if $V_{init}$ encountered:
10:         while not last cryptoAPI
11:             $at \leftarrow API$
12:         end, go to 13
13:     while not EOF:
14:         $post \leftarrow API$
15: $Threshold \leftarrow$ avg($TFIDF(V_{init})$)
16: $d_j = TFIDF(API_j)$
17: for APIs in 'at':        //Calculate the relevant vector
18:     $Q_{relevant} = \sum_{j\ in\ At} d_j$
19: for APIs not in 'at':    // Calculate the irrelevant vector
20:     $Q_{irrelevant} = \sum_{j\ not\ in\ At} d_j$
21: $V_f = V_{init} + \frac{1}{n}(Q_{relevant} - 0.5 * Q_{irrelevant})$
22: for each $q_m \in V_f$:
23:     if $q_m \geq Threshold$:
24:         $V_s \leftarrow q_m$
25: $f(V_s, V_{init}) = V_s \cup V_{init}$
26: $V_b \leftarrow f(V_s, V_{init})$
```

FIGURE 5. The pseudo code of the proposed Dynamic Pre-encryption Boundary Definition (DPBD) technique.

The same procedure was applied on the datasets built using the fixed time-based thresholding of [3]–[5]. To evaluate the efficacy of the DPBD technique, the results were compared with [3]–[5] as shown in Figure 7. The thresholding methods were represented on x-axis while the missed instances (instances that exceeded the boundary) is represented on y-axis. The comparison result suggests that DPBD technique was able to specify the pre-encryption boundaries more accurate than the related works and missed only 8% of crypto-ransomware samples while the related works missed 18% and 29% and 35% of the samples respectively.

To determine the efficacy of the pre-encryption data (DSpre) built using DPBD to build accurate detection models, the comparison was carried out against the datasets built by the related works. The related datasets include DS1, DS2 and DS3 by [3]–[5] respectively. All datasets were used to train same classifiers. The comparison was carried out using accuracy, F1, and precision. Figures 8, 9, and 10 show that the Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), K-Nearest Neighbour (KNN), AdaBoost, Support Vector Machines (SVM) and Multi-Layer

Pseudo Code 2: annotated TF-IDF (aTF-IDF) Technique

Input:
$TR = \{tr_1, tr_2, \ldots\ldots, tr_m\}$; TR denotes a corpus of trace files of the of m programs (benign and ransomware); tr_k is the trace file of the kth instance in the corpus TR.

Output:$Data_{pre}$ the pre-encryption dataset.

```
1: Begin
2: For each $tr_k$ in $TR$:
3:     For each $API$ in $tr_k$:
4:         While $any(V_b)$ is not encountered:
5:             $API_{current} \leftarrow API$
6:             $Annotate(API_{current}, pre)$
7:         $API_{current}++$
8: $F \leftarrow ngram(TR)$
9: For each API-gram($api_k$) in $tr_k$:
10:    $w(api_k) = atf(api_k) \cdot \log\frac{N}{idf(api_k)}$
11: $Data_{pre} \leftarrow aTF-IDF(F)$
12: End
```

FIGURE 6. The Pseudo code of the proposed a TF-IDF technique.

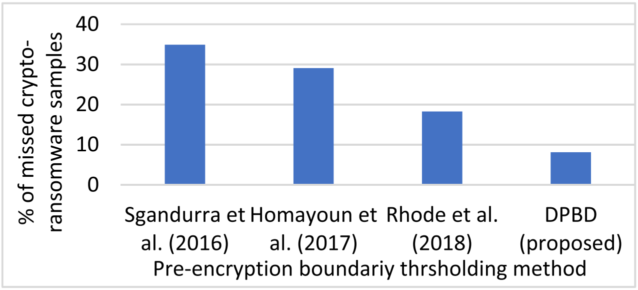


FIGURE 7. Comparison between the accuracy of proposed DPBD with related techniques in determining the pre-encryption boundary.

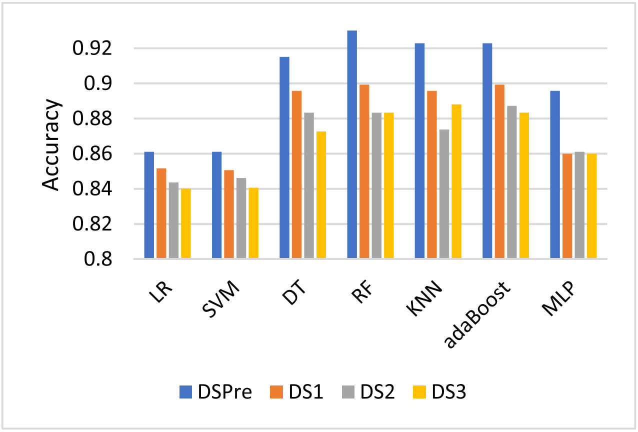


FIGURE 8. The comparison of the classification accuracy between the dataset extracted using proposed DPBD (DSPre) and datasets of the related works DS1, DS2 and DS3.

Perceptron (MLP) trained by the DSpre dataset achieved better classification performance in terms of accuracy, precision and F1.

To evaluate the performance of the proposed annotated Term Frequency-Inverse Document Frequency (aTF-IDF) technique, it was applied on the pre-encryption dataset to extract the features vector for each instance in the corpus. Then, the dataset was divided into training set and testing set using 10-fold cross-validation. Then, the extracted features were used to train several machine learning classifiers including Support Vector Machine (SVM), Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), K-Nearest Neighbors (KNN), AdaBoost and Multi-layer Perceptron (MLP). The testing set was used to measure the classification performance of each classifier based on the extracted features.

Table 3 shows the accuracy results of the classifier trained by dataset extracted using the proposed aTF-IDF method. In terms of classification accuracy, the accuracy level of the classifiers ranged between 0.8648 of SVM and 0.9395 of DT and RF. Similarly, the F1 values were between 0.9275 of SVM and 0.9661 of RF. The results show that precision ranged between 0.8648 of SVM and 0.9414 of DT. For recall, the lowest value was 0.9671 by AdaBoost while the highest value was 1 by SVM. For ROC_AUC, the values ranged between 0.7964 of KNN and 0.889 of adaBoost.

Figures 11, 12 13, 14 and 15 show the comparison results between the proposed aTF-IDF and TF-IDF. It can be observed that the classification accuracy, F1, precision, recall and roc_auc of the proposed aTF-IDF were higher than the traditional TF-IDF for all classifiers.

VI. ANALYSIS AND DISCUSSION

In this paper, the concept of crypto-ransomware early detection based on the dynamic boundaries of the pre-encryption phase was introduced. The DPBD-FE scheme was proposed and implemented to dynamically define the boundaries of pre-encryption phase in the crypto-ransomware lifecycle and extract the pre-encryption data and features based on which, the detection model was trained. The scheme is composed of two components, the Dynamic Pre-encryption Boundary Definition (DPBD) technique and the Features Extraction (FE) technique. The aTF-IDF was proposed as the FE technique. This section discusses and analyses the results of experimental evaluation of the proposed scheme.

As part of DPBD-FE scheme, in this paper, the pre-encryption boundary definition (DPBD) for crypto-ransomware attacks was proposed. Unlike existing solutions that rely on fixed thresholding, the proposed technique adapts a dynamic thresholding approach by building a pre-encryption boundary vector that contains the cryptography related APIs. The entries of this vector were chosen based on their weights calculated using the proposed DPBD. According to comparison results in Figure 7, the DPBD was able to identify the pre-encryption boundary more accurately than the fixed-time thresholding approach employed by the related works. This is attributed to the reliance on the cryptography related APIs to track and identify the starting point of the encryption of each crypto-ransomware sample regardless of the time that sample takes before the encryption. The experimental results in Table 2 and Figure 7 show that, the explicit cryptoAPIs were given scores higher than

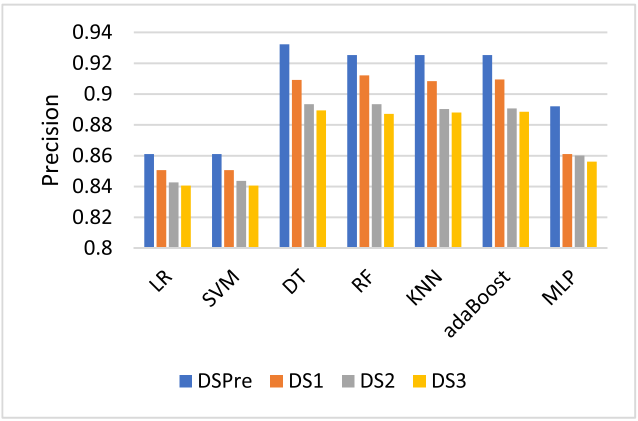


FIGURE 9. The comparison of the classification precision between the dataset extracted using proposed DPBD (DSPre) and datasets of the related works DS1, DS2 and DS3.

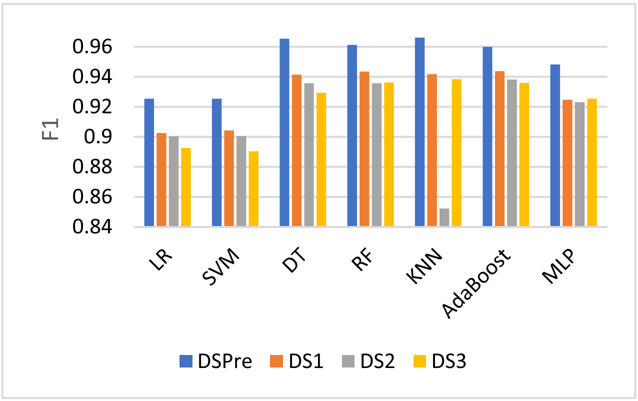


FIGURE 10. The comparison of the classification F1 between the dataset extracted using proposed DPBD (DSPre) and datasets of the related works DS1, DS2 and DS.

other cryptography-related APIs. This indicates that; among all APIs contained within the initAt subset; the proposed DPBD was able to identify the APIs that are more related to cryptography more accurately.

Moreover, figures 8, 9 and 10 show that the classification performance (accuracy, precision and F1) of the algorithms that were trained using the pre-encryption dataset was higher than those of other datasets of related works, i.e. DS1, DS2, and DS3. This implies that the dynamic thresholding of DPBD was able to represent the behavioral aspect of the crypto-ransomware attacks at early phases better than other thresholding techniques. This is attributed to the ability of the DPBD to include more data than other techniques as it tracks the starting point of the encryption process for each instance individually. Therefore, the proposed DPBD makes use of the complementary nature that different ransomware samples might show in the dataset, which gives the proposed DPBD the chance to capture more pre-encryption data. That is, although some ransomware samples start the encryption very early, there are other instances that start the encryption late. Therefore, the dynamic thresholding compensates the

TABLE 3. Experimental results of the aTF-IDF on pre-encryption dataset (extracted using the proposed DPBD technique in the previous section) with different classifiers.

	LR	SVM	DT	RF	KNN	adaBoost	MLP
Accuracy	0.868	0.865	0.939	0.939	0.928	0.925	0.9
F1	0.929	0.928	0.966	0.966	0.959	0.958	0.947
Precision	0.868	0.865	0.941	0.938	0.934	0.927	0.9
Recall	0.996	1	0.979	0.987	0.979	0.967	0.992
ROC-AUC	0.798	0.805	0.885	0.926	0.796	0.889	0.867

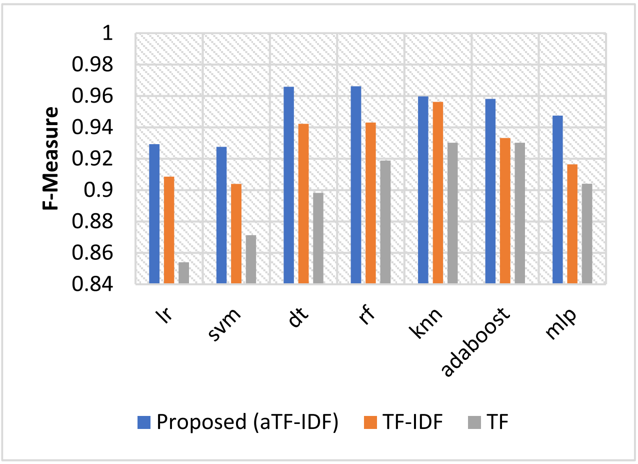


FIGURE 11. The comparison of the F-Measure between the proposed aTF-IDF and the related techniques.

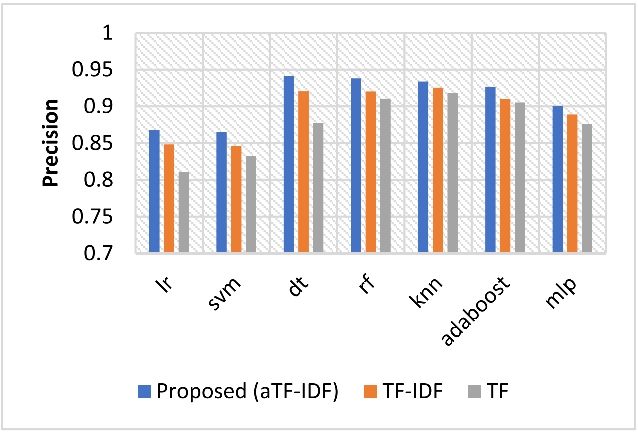


FIGURE 12. The comparison of the Precision between the proposed aTF-IDF and the related techniques.

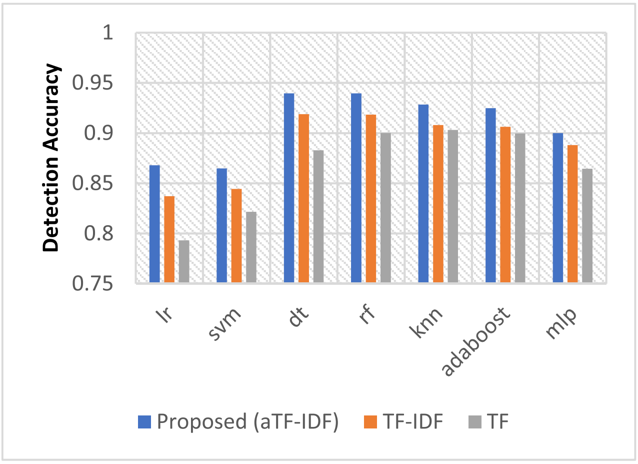


FIGURE 13. The comparison of the Detection Accuracy between the proposed aTF-IDF and the related techniques.

lack of information in the samples that start the encryption early by the information collected by the samples that start the encryption late.

The high the classification performance of all machine learning classifiers shown in Table 3 emphasizes the efficacy of the proposed aTF-IDF technique in identifying the APIs that were closely related to the pre-encryption phase of crypto-ransomware attacks and distinguishing between the informative (attack-specific) APIs and general-purpose ones. This indicates that aTF-IDF was able to highlight the APIs related to crypto-ransomware attacks and down-weights the general-purpose APIs even with the absence of the full observations of attack patterns during the pre-encryption phase. This is attributed to the annotation that aTF-IDF employs, such that it can distinguish the API when it comes before the pre-encryption boundary from the same API when it comes after that. Therefore, the proposed aTF-IDF can easily calculate the TF term according to the pre-encryption data and IDF term according to the full-length data. Consequently, the proposed technique was able to decide whether an API is a general-purpose or attack-specific, given the pre-encryption data only.

The comparison results in figures 11, 12, 13, 14 and 15 show that the proposed aTF-IDF outperformed the traditional TF-IDF in terms of accuracy, F1, precision, recall and roc_auc. This suggests that the proposed technique was able to extract the pre-encryption features more accurately and avoid overweighting the general-purpose APIs. It also emphasizes the importance of calculating the TF term locally based on the pre-encryption data while considering the full-length data when calculating the IDF globally, thanks to the annotation that make it possible for the technique to carry out such calculation.

The time complexity of the proposed aTF-IDF is similar to that of TF-IDF which can be perceived as follows. If N is the number of trace files in the pre-encryption dataset, T is the total APIs in the pre-encryption dataset, then the worst-case time complexity of this algorithm will be O(TN). However, in practice, the number of trace files in which a particular API appears is much less and hence the time taken will be much lower. Therefore, the proposed aTF-IDF can be used efficiently to extract the pre-encryption features in

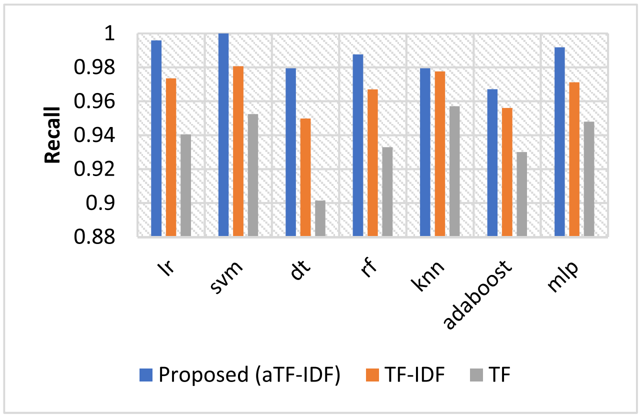


FIGURE 14. The comparison of the Recall between the proposed aTF-IDF and the related techniques.

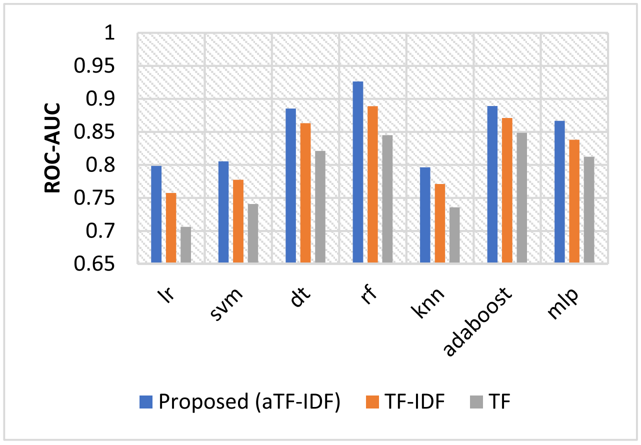


FIGURE 15. The comparison of the ROC-AUC between the proposed aTF-IDF and the related techniques.

real-time, which makes it feasible to be used in real-world deployments. The proposed scheme could either be applied as a full-fledge security measure to protect from malicious software like ransomware or as a part of intrusion detection systems on network and host levels. The scheme will help to early detect the attacks before they start the actual sabotage.

VII. CONCLUSION

In this study, the Dynamic Pre-encryption Boundary Definition and Features Extraction scheme (DPBD-FE) was proposed for crypto-ransomware early detection. The scheme constitutes two components, namely Dynamic Pre-encryption Boundary Definition (DPBD) and Features Extraction (FE). The DPBD builds the pre-encryption boundary vector that contains all cryptography-related APIs by which the boundary of pre-encryption phase in the crypto-ransomware lifecycle was defined. The proposed DPBD technique was able to define the boundary of pre-encryption phase of the crypto-ransomware attacks with accuracy higher than the existing works. The pre-encryption features were extracted from within the pre-encryption using the proposed aTF-IDF technique. With the employment of aTF-IDF, the proposed DPBD-FE scheme was able to identify the features related to the attack patterns at pre-encryption phase and filter out the general-purpose ones. Seven classifiers including Support Vector Machine (SVM), Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), AdaBoost, K-Nearest Neighbours (KNN) and Multilayer Perceptron (MLP) were used to evaluate the classification ability of the pre-encryption data extracted using the proposed scheme. The comparison was also conducted between the proposed techniques in the scheme and the techniques used by exiting works. The results show that the proposed scheme outperformed the techniques used by existing work. Such improvement shows the efficacy of the proposed scheme and techniques in defining the pre-encryption phase more accurately compared to the related works and extract the features that are more related to that phase as well without including general-purpose APIs. The improvement ratio that proposed DPBD achieved in determining the pre-encryption boundary of the attacks was 55%. The results obtained by applying the proposed aTF-IDF show 1%, 0.6%, 0.8%, 0.1% and 4% improvement for precision, accuracy, F1, recall and roc_auc respectively. The proposed scheme could be applied for early detection of other attacks such as malware and intrusion detection.

ACKNOWLEDGMENT

This project was funded by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under grant No. (DF-742-611-1441). The authors, therefore, gratefully acknowledge the DSR technical and financial support.

REFERENCES

[1] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, "Crypto-ransomware early detection model using novel incremental bagging with enhanced semi-random subspace selection," *Future Gener. Comput. Syst.*, vol. 101, pp. 476–491, Dec. 2019.

[2] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, R. Khayami, K.-K.-R. Choo, and D. E. Newton, "DRTHIS: Deep ransomware threat hunting and intelligence system at the fog layer," *Future Gener. Comput. Syst.*, vol. 90, pp. 94–104, Jan. 2019, doi: 10.1016/j.future.2018.07.045.

[3] M. Rhode, P. Burnap, and K. Jones, "Early-stage malware prediction using recurrent neural networks," *Comput. Secur.*, vol. 77, pp. 578–594, Aug. 2018, doi: 10.1016/j.cose.2018.05.010.

[4] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, "Know abnormal, find evil: Frequent pattern mining for ransomware threat hunting and intelligence," *IEEE Trans. Emerg. Topics Comput.*, vol. 8, no. 2, pp. 341–351, Apr. 2020.

[5] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, "Automated dynamic analysis of ransomware: Benefits, limitations and use for detection," 2016, *arXiv:1609.03020*. [Online]. Available: http://arxiv.org/abs/1609.03020

[6] S. Das, Y. Liu, W. Zhang, and M. Chandramohan, "Semantics-based online malware detection: Towards efficient real-time protection against malware," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 2, pp. 289–302, Feb. 2016, doi: 10.1109/tifs.2015.2491300.

[7] N. Nissim, Y. Lapidot, A. Cohen, and Y. Elovici, "Trusted system-calls analysis methodology aimed at detection of compromised virtual machines using sequential mining," *Knowl.-Based Syst.*, vol. 153, pp. 147–175, Aug. 2018, doi: 10.1016/j.knosys.2018.04.033.

[8] E. Kirda, "UNVEIL: A large-scale, automated approach to detecting ransomware (keynote)," in *Proc. IEEE 24th Int. Conf. Softw. Anal., Evol. Reengineering (SANER)*, Feb. 2017, pp. 757–772.

[9] D. Morato, E. Berrueta, E. Magaña, and M. Izal, "Ransomware early detection by the analysis of file sharing traffic," *J. Netw. Comput. Appl.*, vol. 124, pp. 14–32, Dec. 2018, doi: 10.1016/j.jnca.2018.09.013.

[10] S. Jung and Y. Won, "Ransomware detection method based on context-aware entropy analysis," *Soft Comput.*, vol. 22, no. 20, pp. 6731–6740, Oct. 2018, doi: 10.1007/s00500-018-3257-z.

[11] I. Yaqoob, E. Ahmed, M. H. U. Rehman, A. I. A. Ahmed, M. A. Al-garadi, M. Imran, and M. Guizani, "The rise of ransomware and emerging security challenges in the Internet of Things," *Comput. Netw.*, vol. 129, pp. 444–458, Dec. 2017, doi: 10.1016/j.comnet.2017.09.003.

[12] J. Chen, C. Wang, Z. Zhao, K. Chen, R. Du, and G.-J. Ahn, "Uncovering the face of Android ransomware: Characterization and real-time detection," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1286–1300, May 2018, doi: 10.1109/Tifs.2017.2787905.

[13] A. Azmoodeh, A. Dehghantanha, M. Conti, and K.-K.-R. Choo, "Detecting crypto-ransomware in IoT networks based on energy consumption footprint," *J. Ambient Intell. Hum. Comput.*, vol. 9, no. 4, pp. 1141–1152, Aug. 2017, doi: 10.1007/s12652-017-0558-5.

[14] S. D. Yalew, G. Q. Maguire, S. Haridi, and M. Correia, "Hail to the thief: Protecting data from mobile ransomware with ransomsafedroid," in *Proc. IEEE 16th Int. Symp. Netw. Comput. Appl.*, Jan. 2017, pp. 1–8. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85046532213, doi: 10.1109/NCA.2017.8171377.

[15] J. A. Gómez-Hernández, L. Álvarez-González, and P. García-Teodoro, "R-locker: Thwarting ransomware action through a honey file-based approach," *Comput. Secur.*, vol. 73, pp. 389–398, Mar. 2018, doi: 10.1016/j.cose.2017.11.019.

[16] N. Caporusso, S. Chea, and R. Abukhaled, "A game-theoretical model of ransomware," in *Proc. Int. Conf. Appl. Hum. Factors Ergonom.*, vol. 782. Cham, Switzerland: Springer, 2019, pp. 69–78.

[17] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, "Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions," *Comput. Secur.*, vol. 74, pp. 144–166, May 2018, doi: 10.1016/j.cose.2018.01.001.

[18] G. Cusack, O. Michel, and E. Keller, "Machine learning-based detection of ransomware using SDN," in *Proc. ACM Int. Workshop Secur. Softw. Defined Netw. Netw. Function Virtualization*, Tempe, AZ, USA, 2018, pp. 1–6.

[19] D.-Y. Kao and S.-C. Hsiao, "The dynamic analysis of WannaCry ransomware," in *Proc. 20th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2018, pp. 159–166, doi: 10.23919/ICACT.2018.8323682.

[20] N. Hampton, Z. Baig, and S. Zeadally, "Ransomware behavioural analysis on windows platforms," *J. Inf. Secur. Appl.*, vol. 40, pp. 44–51, Jun. 2018, doi: 10.1016/j.jisa.2018.02.008.

[21] A. Cohen and N. Nissim, "Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory," *Expert Syst. Appl.*, vol. 102, pp. 158–178, Jul. 2018, doi: 10.1016/j.eswa.2018.02.039.

[22] A.-R. B. A. Saleh, M. M. Aizaini, P. Y. Adam, S. S. Z. Mohd, and A. A. F. Mohd, "Zero-day aware decision fusion-based model for crypto-ransomware early detection," *Int. J. Integr. Eng.*, vol. 10, no. 6, pp. 11–25, 2018. [Online]. Available: https://publisher.uthm.edu.my/ojs/index.php/ijie/article/view/2828.

[23] S. Song, B. Kim, and S. Lee, "The effective ransomware prevention technique using process monitoring on Android platform," *Mobile Inf. Syst.*, vol. 2016, Mar. 2016, Art. no. 2946735, doi: 10.1155/2016/2946735.

[24] M. M. Ahmadian and H. R. Shahriari, "2entFOX: A framework for high survivable ransomwares detection," in *Proc. 13th Int. Iranian Soc. Cryptol. Conf. Inf. Secur. Cryptol. (ISCISC)*, Rasht, Iran, Sep. 2016, pp. 79–84.

[25] F. Mbol, J.-M. Robert, and A. Sadighian, "An efficient approach to detect torrent locker ransomware in computer systems," in *Proc. 15th Int. Conf.*, Milan, Italy, in S. Foresti G. Persiano Eds. Cham, Switzerland: Springer, Nov. 2016, pp. 532–541.

[26] N. Scaife, H. Carter, P. Traynor, and K. R. B. Butler, "CryptoLock (and drop It): Stopping ransomware attacks on user data," in *Proc. IEEE 36th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2016, pp. 303–312.

[27] R. Moussaileb, B. Bouget, A. Palisse, H. Le Bouder, N. Cuppens, and J. L. Lanet, "Ransomware's early mitigation mechanisms," in *Proc. 13th Int. Conf. Availability, Rel. Secur.*, Dec. 2018, pp. 1–7. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85055288709&doi=10.1145%2f3230833.3234691&partnerID=40&md5=fd3fa38ed1fb15bb45641bb3db029589

[28] M. A. Sotelo Monge, J. M. Vidal, and L. J. García Villalba, "A novel self-organizing network solution towards crypto-ransomware mitigation," in *Proc. 13th Int. Conf.*, 2018, p. 48. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85055254545, doi: 10.1145/3230833.3233249.

[29] Z.-G. Chen, H.-S. Kang, S.-N. Yin, and S.-R. Kim, "Automatic ransomware detection and analysis based on dynamic API calls flow graph," in *Proc. Int. Conf. Res. Adapt. Convergent Syst.*, Sep. 2017, pp. 196–207.

[30] Q. Chen and R. A. Bridges, "Automated behavioral analysis of malware a case study of WannaCry ransomware," 2017, *arXiv:1709.08753*. [Online]. Available: http://arxiv.org/abs/1709.08753

[31] S. Mehnaz, A. Mudgerikar, and E. Bertino. "RWGuard: A real-time detection system against cryptographic ransomware," in *Research in Attacks, Intrusions, and Defense* (Lecture Notes in Computer Science), vol. 11050. Cham, Switzerland: Springer, 2018, pp. 114–136.

[32] K. Cabaj, P. Gawkowski, K. Grochowski, and D. Osojca, "Network activity analysis of cryptowall ransomware," *Przeglad Elektrotechniczny*, vol. 91, no. 11, pp. 201–204, 2015, doi: 10.15199/48.2015.11.48.

[33] K. Cabaj, M. Gregorczyk, and W. Mazurczyk, "Software-defined networking-based crypto ransomware detection using HTTP traffic characteristics," *Comput. Electr. Eng.*, vol. 66, pp. 353–368, Feb. 2018, doi: 10.1016/j.compeleceng.2017.10.012.

[34] A. Tandon and A. Nayyar, "A comprehensive survey on ransomware attack: A growing havoc cyberthreat," in *Data Management, Analytics and Innovation*. Singapore: Springer, 2006, pp. 403–420.

[35] H. J. Chittooparambil, B. Shanmugam, S. Azam, K. Kannoorpatti, M. Jonkman, and G. N. Samy, "A Review of ransomware families and detection methods," in *Recent Trends Data Science Soft Computing*. Cham, Switzerland: Springer, 2019, pp. 588–597.

[36] K. P. Subedi, D. R. Budhathoki, and D. Dasgupta, "Forensic analysis of ransomware families using static and dynamic analysis," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2018, pp. 180–185.

[37] A. Zimba, "Malware-free intrusion: A novel approach to ransomware infection vectors," *Int. J. Comput. Sci. Inf. Secur.*, vol. 15, no. 2, p. 317, 2017.

[38] J. P. Tailor and A. D. Patel, "A comprehensive survey: Ransomware attacks prevention, monitoring and damage control," *Int. J. Res. Sci. Innov.*, vol. 4, no. 15, pp. 116–121, 2017.

[39] L. J. G. Villalba, A. L. S. Orozco, A. L. Vivar, E. A. A. Vega, and T.-H. Kim, "Ransomware automatic data acquisition tool," *IEEE Access*, vol. 6, pp. 55043–55052, 2018, doi: 10.1109/access.2018.2868885.

[40] N. Suditu and F. Fleuret, "Iterative relevance feedback with adaptive exploration/exploitation trade-off," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, Maui, HI, USA, 2012, pp. 1323–1331.P

[41] G. Salton and C. Buckley, "Improving retrieval performance by relevance feedback," *Readings Inf. Retr.*, vol. 24, no. 5, pp. 355–363, 1997.

[42] H. Zhang, X. Xiao, F. Mercaldo, S. Ni, F. Martinelli, and A. K. Sangaiah, "Classification of ransomware families with machine learning based on N-gram of opcodes," *Future Gener. Comput. Syst.*, vol. 90, pp. 211–221, Jan. 2019, doi: 10.1016/j.future.2018.07.052.

[43] J. Stiborek, T. Pevný, and M. Rehák, "Multiple instance learning for malware classification," *Expert Syst. Appl.*, vol. 93, pp. 346–357, Mar. 2018, doi: 10.1016/j.eswa.2017.10.036.

[44] C.-T. Lin, N.-J. Wang, H. Xiao, and C. Eckert, "Feature selection and extraction for malware classification," *J. Inf. Sci. Eng.*, vol. 31, no. 3, pp. 965–992, 2015.

[45] A. Fujino, J. Murakami, and T. Mori, "Discovering similar malware samples using API call topics," in *Proc. 12th Annu. IEEE Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2015, pp. 140–147, doi: 10.1109/CCNC.2015.7157960.

[46] B. Yu, Y. Fang, Q. Yang, Y. Tang, and L. Liu, "A survey of malware behavior description and analysis," *Frontiers Inf. Technol. Electron. Eng.*, vol. 19, no. 5, pp. 583–603, May 2018, doi: 10.1631/Fitee.1601745.

[47] C. Rossow, C. J. Dietrich, C. Grier, C. Kreibich, V. Paxson, N. Pohlmann, H. Bos, and M. V. Steen, "Prudent practices for designing malware experiments: Status quo and outlook," in *Proc. IEEE Symp. Secur. Privacy*, May 2012, pp. 65–79, doi: 10.1109/SP.2012.14.

[48] P. Wang and Y.-S. Wang, "Malware behavioural detection and vaccine development by using a support vector model classifier," *J. Comput. Syst. Sci.*, vol. 81, no. 6, pp. 1012–1026, Sep. 2015, doi: 10.1016/j.jcss.2014.12.014.

[49] J. Stiborek, T. Pevný, and M. Rehák, "Probabilistic analysis of dynamic malware traces," *Comput. Secur.*, vol. 74, pp. 221–239, May 2018, doi: 10.1016/j.cose.2018.01.012.

[50] C. Le Guernic and A. Legay, "Ransomware and the legacy crypto API," in *Proc. 11th Int. Conf.*, Roscoff, France, in vol. 10158. Cham, Switzerland: Springer, Sep. 2016, p. 11.

[51] J. B. Christensen and N. Beuschau, "Ransomware detection and mitigation tool," Tech. Univ. Denmark, Lyngby, Denmark, Tech. Rep., 2017.

[52] A. Ioanid, C. Scarlat, and G. Militaru, "The effect of cybercrime on romanian SMEs in the context of wannacry ransomware attacks," in *Proc. 12th Eur. Conf. Innov. Entrepreneurship*, 2017, p. 307.

[53] S. K. Pandey and B. M. Mehtre, "Performance of malware detection tools: A comparison," in *Proc. IEEE Int. Conf. Adv. Commun., Control Comput. Technol.*, 2015, pp. 1811–1817. http://www.scopus.com/inward/record.url?eid=2-s2.0-84923314531&partnerID=40, doi: 10.1109/ICACCCT.2014.7019422.

[54] M. Alazab, M. Alazab, A. Shalaginov, A. Mesleh, and A. Awajan, "Intelligent mobile malware detection using permission requests and API calls," *Future Gener. Comput. Syst.*, vol. 107, pp. 509–521, Jan. 2020, doi: 10.1016/j.future.2020.02.002.

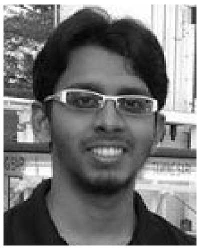
[55] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, and Q. Zheng, "IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture," *Comput. Netw.*, vol. 171, Oct. 2020, Art. no. 107138, doi: 10.1016/j.comnet.2020.107138.

BANDER ALI SALEH AL-RIMY is a senior lecturer at UNITAR International University, Selangor, Malaysia. He received the B.Sc. degree in computer engineering from the Faculty of Engineering, Sana'a University, Yemen, in 2003, the M.Sc. degree in information technology from OUM, Malaysia, in 2013, and the Ph.D. degree in computer science (information security) from the Faculty of Engineering, Universiti Teknologi Malaysia (UTM), in 2019. He is currently a Senior Lecturer with UNITAR International University, Selangor, Malaysia. His research interests include malware, IDS, network security, and routing technologies. He was a recipient of several academic awards and recognitions including the UTM Alumni Award, the UTM Best Postgraduate Student Award, the UTM Merit Award, the UTM Excellence Award, the OUM Distinction Award, and the Best Research Paper Award.

MOHD AIZIANI MAAROF (Member, IEEE) received the B.Sc. degree in computer science from Western Michigan University, Kalamazoo, MI, USA, the M.Sc. degree in computer science from Central Michigan University, Mount Pleasant, MI, USA, and the Ph.D. degree in IT security from Aston University, Birmingham, U.K. He is currently a Professor with the School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia (UTM). He is also the Head of UTM-CSM Cyber Threat Intelligence Lab (CTIL) and a member of the Information Assurance and Security Research Group (IASRG), UTM. His research interest includes information system security.

MAMOUN ALAZAB received the Ph.D. degree in computer science. He is currently an Associate Professor with the College of Engineering, IT, and Environment, Charles Darwin University, Australia. He is a Cyber Security Researcher and a Practitioner with industry and academic experience. His research is multidisciplinary that focuses on cyber security and digital forensics of computer systems, including current and emerging issues in the cyber environment like cyber-physical systems and the Internet of Things, by taking into consideration the unique challenges present in these environments, with a focus on cybercrime detection and prevention. He has more than 100 research articles. He presented at many invited keynotes talks and panels, at conferences and venues nationally and internationally (22 events in 2018 alone). He is an Editor on multiple editorial boards, including the Associate Editor of IEEE Access and an Editor of the *Security and Communication Networks Journal*.

FAWAZ ALSOLAMI received the M.A.Sc. degree in electrical and computer engineering from the University of Waterloo, Canada, in 2008, and the Ph.D. degree in computer science from KAUST University, Thuwal, Saudi Arabia, in 2016. He joined with King Abdulaziz University as an Assistant Professor of computer science. He also published many articles and one monograph. His research interests include artificial intelligence, machine learning and data mining, and combinatorial optimization. He has been the Chairman of the Computer Science Department, King Abdulaziz University, since 2018.



SYED ZAINUDEEN MOHD SHAID (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in computer science from Universiti Teknologi Malaysia (UTM). He is currently a Lecturer with UTM, where he teaches subjects like penetration testing, security programming, exploitation, and other security related subjects. A member of the Information Assurance and Security Research Group (IASRG), he is active in Malware Research. He also does training and consultancy on web security, secure coding, android, and embedded systems. He loves gadgets and enjoys exploring new things related to security.

FUAD A. GHALEB received the B.Sc. degree in computer engineering from the Faculty of Engineering, Sana'a University, Yemen, in 2003, and the M.Sc. and Ph.D. degrees in computer science (information security) from the School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia (UTM), Johor, Malaysia, in 2014 and 2018, respectively. He is currently an Assistant Professor with the School of Computing, Faculty of Engineering, UTM. His research interests include vehicular network security, cyber security, intrusion detection, data science, data mining, and artificial intelligence. He was a recipient of many awards and recognitions, such as the Postdoctoral Fellowship Award from UTM, the Best Postgraduate Student Award from UTM, the Excellence Awards from UTM, the Best Presenter Award from the School of Computing, Faculty of Engineering, UTM, and the Best Paper Awards from many international conferences.

TAWFIK AL-HADHRAMI received the M.Sc. degree in IT/applied system engineering from Heriot-Watt University, Edinburgh, U.K., and the Ph.D. degree in wireless mesh communication from the University of the West of Scotland, Glasgow, U.K., in 2015. He was involved in research with the Networking Group, University of the West of Scotland. He is currently a Senior Lecturer with Nottingham Trent University (NTU), U.K, where he is also a member of the Network Infrastructure and Cyber Security (NICS) Group. His research interests include the Internet of Things (IoT) and applications, network infrastructures and emerging technologies, artificial intelligence, computational intelligence, and 5G wireless communications. He is involved in different projects with industries. He is an Associate Editor of IEEE Access and the IEEE Sensors Journals.

ABDULLAH MARISH ALI received the B.Sc. degree in computer science from Al-Mustansiriya University, Baghdad, Iraq, the M.Sc. degree in computer science from King Abdulaziz University (KAU), Jeddah, Saudi Arabia, and the Ph.D. degree in computer science from Universiti Teknologi at Malaysia (UTM), Malaysia. He is currently an Assistant Professor with the Department of Computer Science, Faculty of Computing and Information Technology, KAU. His research interests include cloud computing, software agents, data mining, and information retrieval.