

Received September 10, 2019, accepted September 25, 2019, date of publication October 7, 2019, date of current version October 17, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2945839

# A Survey on Detection Techniques for Cryptographic Ransomware

EDUARDO BERRUETA<sup>1</sup>, DANIEL MORATO<sup>1,2</sup>, EDUARDO MAGAÑA<sup>1</sup>, AND MIKEL IZAL<sup>1</sup>

<sup>1</sup>Public University of Navarre, Department of Electrical, Electronic and Communications Engineering, Campus Arrosadia, 31006 Pamplona, Spain

<sup>2</sup>Institute of Smart Cities, 31006 Pamplona, Spain

Corresponding author: Daniel Morato (daniel.morato@unavarra.es)

This work was supported by the Spanish MINECO through project PIT (TEC2015-69417-C2-2-R).


**ABSTRACT** Crypto-ransomware is a type of malware that encrypts user files, deletes the original data, and asks for a ransom to recover the hijacked documents. It is a cyber threat that targets both companies and residential users, and has spread in recent years because of its lucrative results. Several articles have presented classifications of ransomware families and their typical behaviour. These insights have stimulated the creation of detection techniques for antivirus and firewall software. However, because the ransomware scene evolves quickly and aggressively, these studies quickly become outdated. In this study, we surveyed the detection techniques that the research community has developed in recent years. We compared the different approaches and classified the algorithms based on the input data they obtain from ransomware actions, and the decision procedures they use to reach a classification decision between benign or malign applications. This is a detailed survey that focuses on detection algorithms, compared to most previous studies that offer a survey of ransomware families or isolated proposals of detection algorithms. We also compared the results of these proposals.

**INDEX TERMS** Computer security, malware detection, ransomware.

## I. INTRODUCTION

In 2018, ransomware dominated the reports on cybercrime, and became a malware threat targeting both businesses and users alike. Some reports describe ransomware not only as a tool of financially motivated criminals, but also for groups suspected of being aligned to the interests of some nations [1]. Ransomware has affected a broad spectrum of companies from manufacturing, transport, and telecommunications industries, financial companies, public law enforcement, and health services [2], [3].

Ransomware extorts users by locking access to computer resources and asking for a monetary payment to recover access. The first reports on ransomware that had a large impact referred to them as lockscreen ransomware. This type of malware locks access to a computer. In some cases, it impersonates law enforcement organisations, asking for the payment of a fine for some illegal activity performed by the user, such as downloading files or browsing child pornography websites. These strains are called police ransomware [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Zahid Akhtar .

Since 2015, police ransomware and lockscreen ransomware in general decayed in popularity and were substituted by a type of ransomware that encrypts user files and asks for a payment to provide the decryption key. This cryptographic ransomware is called crypto-ransomware or cryptoware. Ransomware payments can be carried using social networks [5]. However, most of them carry the payments using cryptocurrencies, and 98% of ransomware families use Bitcoin [6]. Although the reports on their economic impact are hardly accurate [7], each year single strains of crypto-ransomware reach new records of extended multi-national impact. WannaCry and NotPetya attacks in 2017 were estimated to have incurred global costs of more than \$8 billion [8]. During only the first three weeks of 2018, GandCrab ransomware infected more than 50,000 systems. The incurred costs came not only from ransom payments but also from the cessation of business operations, the impact on the public image of affected companies, and insurance consequences.

Not only desktop systems were affected, but also mobile ransomware increased its impact in 2017 [1]. Antivirus companies reported a three-fold increase in ransomware installation packages during the first quarter of 2017 compared to the previous quarter [9], [10]. Since 2018, malware developers

began to use droppers - trojan software that looks like a benign program, but once executed, downloads or extracts the real malware and installs it in a computer [11].

The detection and blockage of crypto-ransomware has been a very active area of research in recent years, because of the massive expansion of this form of malware. Proposals and developments have come from both the industrial sectors and academia. Most detection solutions run local to a user host, working like an antivirus software to detect and block ransomware activity. However, proposals exist that are based on detecting ransomware network activity or blocking network traffic to servers needed by the malware. These detection algorithms are based on a diversity of ad-hoc heuristics and artificial intelligence techniques. The simplest solutions are customised for detecting a single ransomware type; however, more general proposals are capable of the detection of zero-day attacks. Our survey covers the case of crypto-ransomware, and from this point forward, any reference to ransomware refers to a ransomware that encrypts user files.

No previous complete surveys on ransomware detection techniques exist because of the novelty of this type of malware and the fast-paced changing scenario of malware. Most surveys limited their scope to a description of the behaviour of different ransomware families [12]–[14] or the ransomware families were included as a category in a broader study of malware [15]. The description of detection techniques is usually short and only in the form of suggestions and ideas [16], [17] and not formal algorithms. The most complete surveys on the topic are those of Eze et al. [18] and in particular of Al-rimy et al. [19]. Although Eze et al. [18] cites several research papers on ransomware detection in a simple classification, it does not provide a complete review of the literature, but only a two-page introduction to the topic. Al-rimy et al. [19] presents an extensive classification on ransomware behaviour and an up-to-date classification of detection algorithms in the research literature. However, he does not provide a description of the input parameters and classification results provided by these proposals. Further, he covers the broad topic of ransomware without providing sufficient detail on the detection algorithms, which is the sole topic of our review.

In this survey, we reviewed 50 proposals for the detection of cryptoware activity. Most came from academia, but we also added the description of existing commercial products based on the available public information. We present a classification based on the behavioural characteristics extracted from a running system and the mathematical tools used in their analysis: machine learning (ML)-based and threshold based.

The techniques for ransomware detection are based on distinguishing ransomware action from benign software. In this study, we classified ransomware action as different detection techniques. This is a different classification from those presented in the literature. An in-depth analysis of ransomware characteristics is vital to create an accurate detection algorithm. We describe the characteristics of 48 different ransomware families from the analysis of 64 collected samples.

We extracted these characteristics from running the actual samples. The survey papers in the literature describing ransomware action had no access to such a large database and did not offer first-hand analysis. We classify these families and relate them to the proposed detection techniques.

There are no common metrics of accuracy and performance in ransomware detection. Some studies with a new detection algorithm compared their proposals with previous ones, but not all of them, and only to the extent of the ransomware families they targeted. We describe and compare the results of all these proposals, and we complete this paper with a description of unsolved issues.

The remainder of this paper is structured as follows: Section II details the characteristics of the different ransomware families from the point of view of the design of detection algorithms. Section III offers a classification and description of the input data that the detection algorithms extracted from ransomware. Section IV groups the detection algorithms based on the logic of how to use the input data to obtain a classification decision. We also describe the main aspects of these algorithms. Section VI presents the detection and performance results offered in the literature. Section VII discusses open issues in crypto-ransomware detection. Section VIII concludes the paper.

## II. CHARACTERISTICS OF RANSOMWARE BEHAVIOUR

We obtained more than 60 samples of different ransomware programs during a six-year period. We tested their behaviour in a virtualised environment, analysing their dynamic behaviour. Based on this analysis, we established a small number of actions that ransomware performs and classified ransomware based on the selected actions performed and their implementation. These actions have been described in other studies [12], [15]; however, we take the point of view of the detection algorithm and the characteristics of its use, which is a different approach. The details of this behaviour are critical in the design of a ransomware detection algorithm.

Figure 1 presents the five steps of ransomware activity. We briefly name and describe them as follows:

- 1) **Infection:** A victim's computer is infected. The attack vectors are shared with other types of malware. They commonly reach the victim as e-mail-attached files executed by the user [20]. They can also be downloaded from infected web pages, taking advantage of web browser vulnerabilities [20]. Finally, some ransomware strains implement a worm-like behaviour capable of its distribution in a Local Area Network (LAN) [21].
- 2) **Contact Command and Control (C&C) servers:** The malware contacts a C&C server to obtain or store the



FIGURE 1. Steps in ransomware activity.

encryption key. The server can be located using statically configured Internet Protocol (IP) addresses, static Domain Name System (DNS) names, or dynamically generated DNS names.

- 3) Encryption key management: The encryption key is obtained from C&C servers or locally generated and then stored on remote servers. In the last case, Step 2 can take place after Step 3. The malware can also generate and store the key locally, encrypting it using the remotely obtained key. The main action in this step is the local management of encryption keys.
- 4) Data encryption: The main task of cryptoware is the encryption of user files, which occurs in this step. It encrypts and deletes local files, but it can also affect files and volumes mounted using a network file sharing protocol. It can also erase backup files or volumes (for example, shadow copies in Microsoft Windows systems) and modify the computer startup sequence to run again if the computer is rebooted.
- 5) Extortion: The ransomware requests the payment of a ransom to decrypt the files. It explains to the user what has occurred and what must be done to recover the files. It can show a new window with this information, include text files in every directory containing the explanation, or change the desktop background.

Table 1 shows the ransomware families and how they implement each step from 2 to 5. The detection techniques take advantage of known ransomware behaviour during one or more of the above-mentioned steps. In the following sections we detail the activities taking place in each step and the corresponding entries in Table 1.

### A. INFECTION

The infection step (Step 1) is carried out following the same procedures as any other malware. The main infection vectors are:

- Spam: The malware reaches the user through unsolicited e-mail. It contains the malware as an attached file. The content and subject of the e-mail try to be related to personal or work activities (requests from friends, rebates, invoices).
- Corrupted web pages: A web page containing benign files is hacked by the malware author. The files offered for download (software updates, documents) are substituted for the malware. The user downloads and executes the malware while believing it is a benign document.
- Vulnerabilities: The hacker benefits from vulnerabilities in the user operating system or web browser to deliver and run the malware to the host.
- Phishing: The content of an e-mail, web page or the interlocutor on a phone call impersonate a company, friend, colleague or public service to take the user to a fake web page and to download and run the malware.

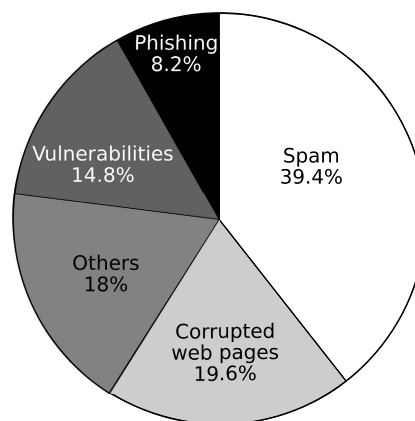


FIGURE 2. Ransomware infection vector.

Figure 2 shows the distribution of the infection vector for ransomware strains described in Table 1 [22]–[25]. We refer to the literature [26], [27] for further detail.

### B. CONTACT C&C SERVERS PREVIOUS TO DATA ENCRYPTION

Step 2 comprises the actions a ransomware performs which require Internet access and are previous to the data encryption phase. It contacts a C&C server or a cluster of servers. A C&C server is an Internet host or set of hosts that manage malware actions, distribute commands, and collect information from victims [28].

This is not a necessary step previous to the encryption phase. Ransomware can generate a local key for a symmetric key encryption algorithm, encrypt target files, and then store the key on a C&C server. It can also locally store the encryption key and never contact an external server. In that case, the key is encrypted using a public key from an asymmetric pair so that only the attacker can decrypt it using the corresponding private key. Therefore, the host needs only to contact the hacker when the user asks for the decryption key. These attack patterns are vulnerable because the decryption key is in the victim's host in some moment and form (in local disk or maybe only in RAM before it is encrypted).

Ransomware strains such as CTBLocker and DMALocker do not contact servers previous to the encryption phase. However, we obtained 17 ransomware strains that contacted external servers previous to the data encryption phase. Some of them obtained a key from an asymmetric pair from a C&C server and encrypted the files using this key. This method ensures that the decryption key is never stored in the infected host; therefore, it cannot be obtained by the antivirus program.

Ransomware actions during this phase imply the generation of network traffic contacting external servers, not requested by user actions. This network traffic is used by some analysis techniques to detect a ransomware in action. We considered the following four categories of ransomware behaviour (Figure 3) based on the actions taken in this step.

TABLE 1. Examples of ransomware families and characteristics.

Year	Family	C&C	Encryption	Change extension	Infection	Platform	
2013	CryptoLocker	Static domain	Symmetric. AES encryption	Yes	Spam & Corrupted web pages	Windows	
	CryptoDefense	Static domain	Asymmetric	No	Spam & Phishing	Windows	
	SympLocker	Static domain	Symmetric. AES encryption	Yes	Fake apps	Android	
	CryptoWall	Static domain	Asymmetric. RSA-2048 bits	Yes	Spam & malicious ads or sites	Windows	
	Virlock	Static domain	Asymmetric. RSA-2048 bits	Yes	Encrypted files	Windows	
2014	TeslaCrypt	Static domain & DGA domains	Symmetric. AES-256-CBC	Yes	Spam & Phishing	Windows	
	CTBLocker	No contact to C&C	Symmetric. AES encryption	Yes	Spam	Windows	
	CryptVault	Static domain	Asymmetric. GnuPG software	Yes	Spam	Windows	
	Tox	Static domain	Symmetric. AES encryption	Yes	Spam	Windows	
	TorrentLocker	Static domain & DGA domains	Symmetric. AES encryption	Yes	Spam	Windows	
	DMALocker	Static domain & no contact to C&C	Symmetric. AES encryption	Yes	Compromised sites	Windows	
	Linux.Encoder	Static domain	Asymmetric. RSA algorithm	Yes	Plugin vulnerabilities	Linux	
	Chimera	Static IP Address	Symmetric. AES encryption	Yes	Phishing	Windows	
	Locky	DGA domain	Symmetric. AES encryption	Yes	Spam	Windows	
	SamSam	Static domain	Asymmetric. RSA-2048 bits	Yes	Vulnerable servers	Windows	
	Petya	Static domain	Symmetric. AES encryption	Yes	Fake software. Worm	Windows	
2015	Xorist	Static domain	Symmetric. TEA cipher	Yes	Spam	Windows	
	Jigsaw	No contact to C&C	Symmetric. AES encryption	Yes	Spam	Windows	
	KeRanger	Static domain	Asymmetric. RSA algorithm	Yes	Fake apps	MacOS	
	Cerber	Static domain & no contact to C&C	Symmetric. AES encryption	Yes	Spam & fake software	Windows	
	CryptXXX	Static domain	Symmetric. AES encryption	No	Spam	Windows	
	Enigma	Static domain	Symmetric. AES encryption	Yes	HTML attachments	Windows	
	Bart	No contact to C&C	Symmetric. AES encryption	Yes	HTML attachments	Windows	
	Fsociety	Static domain	Symmetric. AES encryption	Yes	Spam & malicious URLs	Windows	
	Raa	Static domain	Symmetric. AES encryption	Yes	Spam	Windows	
	Anubis	Static domain	Symmetric. AES encryption	Yes	Spam	Windows	
	Matrix	Static domain	Symmetric. AES encryption	Yes	Spam	Windows	
	Locky	DGA domain	Symmetric. AES encryption	Yes	Spam	Windows	
	Satan	Static domain	Symmetric. AES encryption	Yes	Worm (EternalBlue)	Windows	
	KillDisk	No contact to C&C	Symmetric. AES encryption	Yes	Spam	Linux	
	Sage	No contact to C&C	Symmetric. AES encryption	Yes	Spam	Windows	
	2016	CryptoShadow	Static domain	Symmetric. AES encryption	Yes	Spam	Windows
		DoubleLocker	No contact to C&C	Symmetric. AES encryption	Yes	Fake apps	Android
CryptoShield		Static domain	Symmetric. AES encryption	Yes	Compromised sites	Windows	
Patcher		No contact to C&C	Symmetric. AES encryption	Yes	Fake apps	Windows	
Revenge		No contact to C&C	Symmetric. AES encryption	Yes	Compromised sites	Windows	
BTCWare		No contact to C&C	Symmetric. RC4 encryption	Yes	RDP & Spam	Windows	
Erebus		Static domain	Symmetric. AES encryption	Yes	Malvertising	Linux	
WannaCry		Static domain	Symmetric. AES encryption	Yes	Worm (EternalBlue)	Windows	
WannaLocker		Static domain	Symmetric. AES encryption	Yes	Fake game plugin	Android	
Crysis		No contact to C&C	Symmetric. AES encryption	Yes	RDP	Windows	
NotPetya		No contact to C&C	Symmetric. AES encryption	Yes	Worm (EternalBlue)	Windows	
GlobeImposter		No contact to C&C	Asymmetric. RSA-2048	Yes	Software & compromised websites	Windows	
BadRabbit		No contact to C&C	Symmetric. AES encryption	Yes	Compromised web sites	Windows	
Locker		Static domain	Symmetric. AES encryption	Yes	Spam & Compromised web sites	Windows	
2017	GandCrab	Static domain	Symmetric. AES encryption	Yes	Fake software crack sites	Windows	
	Hermes2.1	Static domain	Symmetric. AES encryption	Yes	GreenFlash Sundown Flash Player	Windows	
	Retwyware	Static domain	Symmetric. AES encryption	Yes	Spam	Windows	
	Ryuk	No contact to C&C	Symmetric. AES encryption	Yes	Directed attacks to businesses	Windows	
	Katyusha	Static domain	Asymmetric. RSA-2048	Yes	Worm (EternalBlue)	Windows	
	Scarab	No contact to C&C	Symmetric. AES encryption in CBC mode	Yes	Spam	Windows	
	LockerGoga	No contact to C&C	Asymmetric. RSA-4096	Yes	Worm (PsExec tool)	Windows	

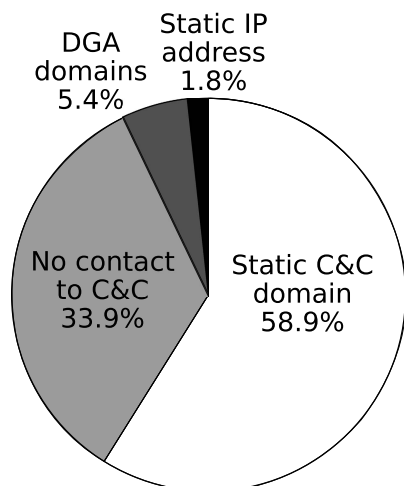


FIGURE 3. Ransomware C&C contact methodologies.

#### 1) NO CONTACT TO C&C

For file encryption, the ransomware can use a public key from an asymmetric pair, being the private one known only to the attacker. However, public key encryption is usually slower than encryption using a shared-key algorithm. Some ransomware strains create a key for a symmetric encryption algorithm and store the key within the file (a different key for each file). In this scenario, the only information encrypted using public key cryptography is the key for the symmetric key algorithm [29]. File decryption requires the private key to decrypt the symmetric key used for encrypting the file. No communication is required previous to the data encryption phase, because the public key is included within the ransomware binary.

Families taking this approach are CTBLocker, DMALocker Jigsaw, Bart, Spora, KillDisk, Sage2.0, Patcher, Revenge, BTCWare, Crysis, NotPetya, GlobeImposter, Ryuk, Scarab, LockerGoga, and BadRabbit [23], [29]–[37]. These ransomware strains cannot be detected by network traffic previous to the loss of data files.

#### 2) STATIC IP ADDRESSES OF C&C SERVERS

The ransomware knows the static IP address used by the C&C server. It can be a list of addresses. Some of them can continue working even after a firewall blocks access to addresses that have been detected as malware related. Ransomware using static IP addresses is only effective in a zero-day scenario. As the C&C server addresses are discovered (from network traffic or from the ransomware binary), they can be added to firewall blacklists, rendering the malware ineffective.

Chimera ransomware [38] shows this behaviour.

#### 3) STATIC DNS DOMAINS FOR C&C SERVERS

Instead of knowing the IP address of a C&C server, the ransomware knows its DNS name (or a list of names). Similar to the case with static IP addresses, this method is vulnerable to the analysis of the ransomware binary. The list of DNS names

can be extracted from the binary file and added to black lists in the DNS servers. This was the case in the infamous WannaCry attack [22].

This is a very common approach used by more than 30 ransomware families we analysed (Table 1).

#### 4) DYNAMICALLY GENERATE DOMAIN NAMES

The ransomware does not contain a list of domain names to locate a C&C server but a Domain Generation Algorithm (DGA). This algorithm creates pseudo-random domain names that it tries to resolve to obtain the IP address from a C&C server. The attacker knows the algorithm and the seed used in the random generation and can predict the names it will check and reserve those names in advance. Since the names change, blacklisting a list of names becomes infeasible.

The drawback in this strategy from the attacker's point of view is the long time it takes for the ransomware to generate an active domain name and to contact a server. This time provides a window of opportunity for ransomware detection previous to the phase of data encryption. TeslaCrypt, TorrentLocker and Locky [39]–[41] are three ransomware families that use a DGA. Some ransomware detection proposals which focused on detecting a random pattern in DNS requests to detect the activity of these families will be described here.

### C. ENCRYPTION KEY MANAGEMENT

The type of encryption algorithm used by a ransomware and the key management strategy determine the presence of the decryption key in a user's machine. Anti-ransomware software running on an infected host can detect the presence of the key and retain it. However, by using asymmetric cryptography, the ransomware can keep the decryption key away from the user's host. Therefore, both the type of encryption algorithm and the key management strategy are critical in anti-ransomware software design. In this section, we describe the options on both axes. Their implementation in the different ransomware families is summarised in Table 1.

#### 1) ENCRYPTION ALGORITHMS

Both symmetric and asymmetric key algorithms are used in different types of ransomware. Symmetric key algorithms such as the Advanced Encryption Standard (AES) provide faster encryption than asymmetric key algorithms such as RSA-2048. Using a simpler algorithm, the ransomware can finish its encryption task earlier, thereby reducing the time interval during which it can be detected by the user. Simpler encryption also allows the ransomware to consume less computing resources, which can keep it undetected by the user while it is encrypting files.

The main disadvantage of symmetric key algorithms is that the decryption key is the same as the encryption key. Therefore, the decryption key is present in the host computer while the ransomware is encrypting files. The ransomware

removes this key from the host when it has finished using it in the encryption phase. It can keep it in the host but only after encrypting it using a public key whose private key is only known by the attacker. In any case, an anti-ransomware software running on a user's host can detect the presence of the encryption key and obtain it.

## 2) KEY GENERATION AND DISTRIBUTION

Ransomware can obtain the key from C&C servers or generate it locally. In these scenarios, when the ransomware generates cryptographic keys, it uses system function calls that provide this service (e.g. the `CryptGenRandom` function in a Windows system). Anti-ransomware software can detect this function call and stop the calling process. However, as benign software may also use these function calls, it is a better strategy to store any key generated in the system in case it was used by malware. However, this can pose confidentiality problems to benign software because the keys have been retained.

Ransomware strains DMA Locker, Chimera, Locky, Cerber, Bart, BTCWare, WannaCry, NotPetya, and BadRabbit [22], [23], [25], [42]–[46] are known to use the `CryptGenRandom` function for the generation of encrypting keys.

The safest procedure for ransomware is to use a public key for file encryption. The pairing private key should only be known by the attacker. Ransomware can contain the public key in its binary with the disadvantage of being a common key for any infected user. In a smarter strategy, the public key is user-specific and obtained from C&C servers previous to the encryption phase. The attacker keeps the corresponding private key in its servers and can release it after ransom payment. Using this strategy, key retrieval is impossible because the decryption key never leaves the attacker's computers.

## D. DATA ENCRYPTION

The main task in a ransomware lifecycle is encrypting the content of user files, rendering them unusable unless the user pays a ransom to obtain the decryption key. During the encryption phase, the user can notice ransomware activity as the files are being modified; therefore, to be effective, this task must be completed in the shortest amount of time. However, fast encryption requires CPU resources, which can also betray the ransomware. Therefore, there exists a tradeoff between a high CPU load and longer encryption time, both of which help ransomware detection.

To reduce the CPU load and encryption time, a ransomware can encrypt only a limited number of bytes in each file. The rest of the file is untouched. Although part of the user's information is unencrypted in the file, most file formats become corrupted and are difficult to recover, even the unencrypted part.

Anti-ransomware monitoring software can monitor user files and detect file content corruption. By checking only the first bytes in a file (the magic bytes), a file type can usually be detected. The ransomware encrypts the entire file, therefore changing the magic bytes. However, famous strains like

Cerber [47] keep the magic bytes unencrypted, hampering detection techniques based on the checking of these bytes.

As the final phase in file modification, some ransomware strains change the file extension, while others maintain the original extension. These changes can be used to detect ransomware activity. Table 1 details some of these behaviours in the data encryption step.

## E. EXTORTION

To reveal itself to the user, the malware asks for payment of a ransom. The most frequent technique is the creation of text files, HTML documents, or image files inside the directories where the files were encrypted. In these files the hacker informs the user as to what has occurred and what the user must do to recover one's files. Some strains of ransomware use system calls to change the computer desktop background or lock access to the computer and show only the ransom information.

The extortion phase does not always take place once all the files are encrypted. Ransomware can create information files in each directory where it has encrypted files when all the files in the directory have been corrupted. Therefore, the detection of ransomware activity based on this method of informing the user is feasible, although the ransomware detection is usually too late to be effective.

## III. INPUT INFORMATION FOR RANSOMWARE DETECTION

The techniques proposed by the industry and academia for ransomware detection extract information from the suspected malware before it runs (or while it is running). This information is used for the classification as benign or malign software.

In this section, we present a classification of the parameters and information extracted from ransomware behaviour. They are related to the different ransomware activities described in Section II. We grouped more than 16 parameters related to previous activities into 3 sets in the first classification level. We differentiated information extracted locally to the infected machine (static or dynamic information) from information extracted from network traffic (Figure 4). The three categories discussed in the following subsections are:

- 1) Local static: The information is extracted from the malware binary before it runs and can be obtained before launching the malware.
- 2) Local dynamic: The data is extracted while the program is running based on the actions it takes on the infected computer.
- 3) Network based: The information comes from the network traffic created by the running malware.

Table 3 shows a list of scientific references on ransomware detection and the types of input information each one requires. In the following subsections we describe the different levels of classification in the three above-mentioned categories.

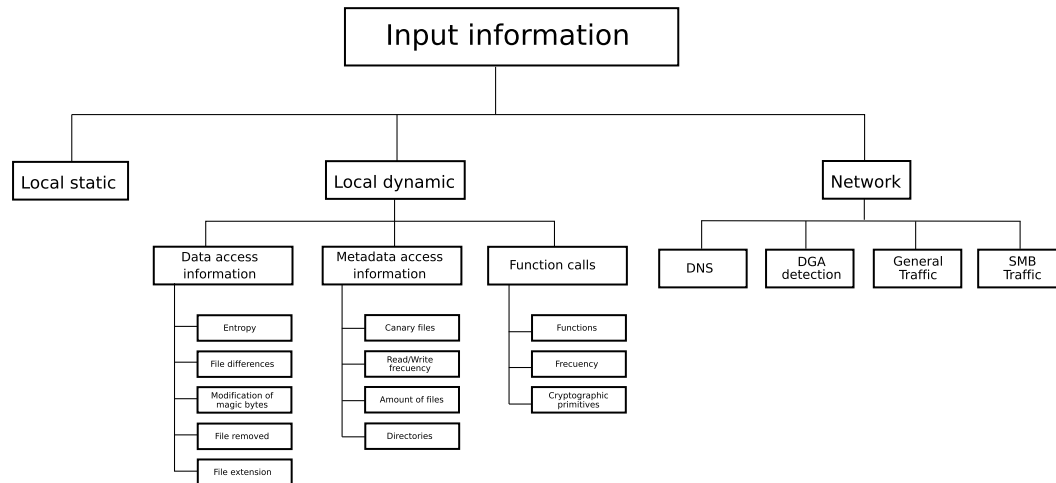


FIGURE 4. Ransomware detection tool classification based on input data.

**A. LOCAL STATIC INFORMATION**

A detection algorithm based only on local static parameters is capable of detecting malware before it runs. An effective algorithm based only on local static parameters is the most effective and avoids any loss of user data.

A common technique used in commercial antivirus software is to obtain the local static parameters from the analysis of the program binary. However, some ransomware strains [24] use code obfuscation techniques or a polymorphic behaviour [48], which hinders detection. The static information obtained from the files is related to text strings or function calls.

1) TEXT STRINGS

Common strings found in ransomware binaries are “ransom”, “bitcoin”, or “encrypt”. It can also contain well-known domain names or IP addresses. The anti-malware software can search for keywords or set phrases [49], [50]. It is usually complemented with a deeper analysis of static or dynamic parameters because the method is prone to false positive alerts.

2) FUNCTION CALLS

The most common function calls found in ransomware programs are related to cryptography algorithms (key generation, encryption, and decryption) and file access. Binary inspection can detect the use of these suspect function calls. They can be functions from well-known dynamic system libraries or statically linked libraries.

**B. LOCAL DYNAMIC INFORMATION**

Local dynamic parameters are extracted once the malware is running. They present the disadvantage of the requirement to run untrusted software. However, dynamic parameters are more difficult to obfuscate because the ransomware has no option but to take action. For example, it can avoid using

TABLE 2. Local dynamic information used in different detection algorithms.

Data access information	Metadata information	Functions calls
Mbol et al. [51] Lee et al. [52]	Moore [53] Moussaileb et al. [54]	Vinayakumar et al. [55], Alam et al. [56], Al-rimy et al. [57], Chen et al. [58]
Scaife et al. [59], Shukla et al. [60], Kharraz et al. [61], Kirda [62], Paik et al. [63], Song et al. [64]	Feng et al. [65], Lu et al. [66], Hasan et al. [67], Zhang et al. [68], Yang et al. [69]	
Sgandurra et al. [50], Continella et al. [70], Ahmadian et al. [71], Khashif et al. [49], Mehnaz et al. [72]		

system calls for key management or use a new encryption algorithm; however, it cannot avoid opening, reading, and writing to files.

Dynamic information can be statistical in nature; therefore, it requires collecting samples of ransomware actions during a certain interval of time. During this period, the malware is running free and can perform irreversible destructive actions. Therefore, the data collection phase must be short, and the blocking decision must be made before further loss of user files. However, it cannot be too short as to provide erroneous input data to the detection algorithm and thus render a wrong decision. Ignoring real malware (false negative) and blocking benign software (false positive) must be considered as fatal algorithm errors.

We grouped the local dynamic parameters into three categories (Table 2):

- 1) Data access information: These parameters are related to the modification of the content in user files.
- 2) Metadata access information: They measure the actions taken by the ransomware on user files, not the content of the files, but how and when the files were modified.
- 3) Function calls: They measure the actual library or system functions called by the suspect process.

## 1) DATA ACCESS INFORMATION

While a malware is running, analysis software can extract information about how the data in user files are being accessed and modified. This extracted information is based on the detection of new encrypted content.

### *a: MODIFICATION OF MAGIC BYTES OR FILE FORMAT MODIFICATION*

Frequently, a file format can be recognised by a short number of bytes at the beginning of the file (magic bytes). A program that modifies the content of a file (e.g. editing a text document) does not change the file format. A program that encrypts a file overwrites its content, changing the bytes that characterise the file format.

An encryption action can be detected by watching the modification of the first bytes in modified files. This technique is prone to false positives because common file formats modify this section of file content.

### *b: ENTROPY*

Statistically, encrypted data looks like uniformly distributed random bytes. When a file is written or overwritten, the new content can be checked for randomness. This can be accomplished by intercepting system calls used for file access. A frequently used metric is entropy. Equation 1 shows a common definition of entropy in file content where  $P_{B_n}$  is the probability of byte value  $n$  appearing in a file, although other options exist [52].

$$e = \sum_{i=0}^{255} P_{B_i} \log_2 \frac{1}{P_{B_i}} \quad (1)$$

There is no absolute entropy value that differentiates encrypted from non-encrypted content. An ad-hoc threshold for entropy value must be determined. False positive alarms may occur if this parameter alone is used in ransomware detection. This is not only owing to benign encryption tools, but also because many file formats use compression techniques. A compressed file may be statistically similar to an encrypted file and lead to an incorrect conclusion.

Entropy computation is based on the analysis of the entire file, taking an important toll on CPU usage. It can also add stress to the disk if the anti-malware software analyses the content of all recently modified files instead of intercepting system calls.

### *c: FILE DIFFERENCES*

The encrypted version of a file should not provide any information regarding the unencrypted data. When a file is overwritten, the new content can be compared to the old content. This comparison requires intercepting the file access system calls. If the file is substantially modified, it can be a suspect action because it may be owing to the encryption of the file. If a ransomware writes the encrypted data to a different file, it is much more difficult to correlate read data to written data for comparison purposes.

File differences are prone to trigger false positive alarms when a benign program substantially modifies a file. These differences can be combined with entropy computation because both parameters require the analysis of file content.

### *d: FILES OVERWRITTEN OR REMOVED*

Any modification of file content can be treated as possibly dangerous. When a file is deleted or its content is modified, user data are lost. File access system calls can be intercepted, and these two situations can be detected. Most benign software modifies files; therefore, this parameter cannot be used alone. However, this is an action that ransomware must perform. To be effective at asking for a ransom, the original user data must be removed.

### *e: FILE EXTENSION MODIFICATION*

Most ransomware strains use a specific file extension for encrypted files. They either create new files with a new file name or they overwrite the original file and rename it or change the file name extension.

The file name extension can be checked using a list of well-known extensions used by ransomware. This is a simple procedure, not CPU intensive, and commonly used by commercial anti-virus software.

A file name extension can be considered as file metadata. We consider a file name as data because it is part of the data provided by the user at file creation and must be protected against ransomware actions. For example, the file creation time is not actively provided by a user and usually does not require protection.

## 2) METADATA ACCESS INFORMATION

We consider access to file metadata as any action that does not require file content. This information can be obtained by intercepting file system calls, but it does not require the analysis of file data; therefore, less CPU time is required for its computation.

### *a: FREQUENCY OF FILE SYSTEM CALLS*

All analysed ransomware strains attempt to encrypt as many files as possible in the shortest amount of time. They produce a pattern of very frequent file system operations reading, writing, deleting, or renaming files. A time series with these file operations can be obtained and analysed to distinguish a file access pattern characteristic of ransomware action.

The analysis of this type of information is common to many anti-ransomware proposals. Its main drawback is the requirement of deciding a frequency or pattern of file access that distinguishes between benign and malign software. This can be accomplished by an analysis of previous typical user file access patterns. However, even though the common user behaviour may be very different from the ransomware behaviour, there exist situations where in a short time they may be very similar. For example, when a user compresses a directory of files and removes the original files, the entire



operation may be taken as ransomware action from the point of view of frequency of file operations and file deletions.

#### *b: NUMBER OF ACCESSED FILES*

Ransomware tries to access all user files in a volume, or at least those documents whose file type indicates that they are user files (for example, a user would not typically pay a ransom to recover encrypted operating system files). The number of files accessed by a malware is large even in a short time. In comparison, user actions are typically slower and do not alter many files in a few seconds. However, there are exceptions; for example, a user may work on a batch of files using editing software. There are also exceptions to ransomware accessing many files in a short time; for example, if it is encrypting a very large file or it attempts to hide its activity by slowing down its actions.

#### *c: DIRECTORIES ACCESSED*

Ransomware encrypts user files from any directory and accesses a large amount of paths. It commonly begins by listing the entire tree of files in a volume or at least those files with a set of types. It is uncommon that benign applications access files from a large quantity of different directories; therefore, this behaviour can be used to differentiate between benign and ransomware activity.

#### *d: CANARY FILES*

Some anti-malware tools create files in the user directories. These “canary” files are watched by a monitoring tool. The user is not expected to access these files. If these files are modified, it may be the result of destructive malware; therefore, if the tool identifies the process that altered the file, it can alert the user or block the malware.

To achieve early ransomware detection, an anti-malware tool must spread canary files throughout most of the directories containing user files. Therefore, not only does it clutter user directories, but it requires monitoring the modification of a large number of files.

### 3) FUNCTION CALLS

The most common actions taken by ransomware are related to encryption and key generation. System functions or library functions are used for these tasks. Ransomware detection software can monitor the frequent use of a set of sensible functions and mark a process as dubious.

Cryptographic keys can also be recognised in process memory owing to its structure. Monitoring software can inspect process memory searching for keys and alert the user or keep a copy of the key. This last option is only useful in the case of symmetric key encryption or public key encryption, where the decryption key is present in the system. Using public key encryption, the private key can be isolated from the user host; therefore, although a key may be located in memory, and it may be used to raise an alarm, the decryption key would not be available.

### C. INFORMATION EXTRACTED FROM NETWORK TRAFFIC

Common infection patterns require Internet access (e-mail-attached files and malicious websites). Some ransomware samples do not require further network access once they infect a host. However, most ransomware strains require Internet access to operate. They retrieve keys from C&C servers or they store locally generated keys there.

Network traffic can be obtained at an infected host or at the local network Internet access link. Anti-malware software can analyse this traffic and detect ransomware action. If the action is previous to the data encryption phase, it can block the ransomware before it takes destructive actions. This is most effective if it blocks the ransomware while attempting to obtain an encryption key from C&C servers because this prevents the ransomware from encrypting files.

After reviewing the proposals in the literature, we grouped the information obtained from traffic analysis into three groups. Two are based on the analysis of DNS queries, while the third encompasses a large number of parameters that can be extracted from network traffic.

#### 1) DETECTION OF DNS QUERIES FOR STATIC NAMES

Ransomware software can contain the C&C server IP address in its binary; however, this is an uncommon practice as it makes it vulnerable to network filtering once these addresses are discovered. A more common practice is the dynamic location of a C&C server based on the resolution of a domain name hardcoded in the ransomware binary. The name to address relation can be changed dynamically by a hacker to avoid IP filtering.

Table 1 shows 32 ransomware families that locate C&C servers based on the resolution of a DNS name. This detection technique can block ransomware before it destroys files if the name resolution occurs before the encryption phase. However, it requires the knowledge of the domain names to block, which are extracted from the analysis of the ransomware in action in other hosts or the analysis of its binary file. Therefore, it cannot be used in a zero-day scenario.

#### 2) DETECTION OF DNS QUERIES FOR DYNAMICALLY GENERATED NAMES

For ransomware strains that use a DGA, blocking the resolution of certain domain names is useless. The ransomware can attempt the resolution of dozens of pseudo-randomly generated domain names before it finds a valid one. However, the characteristic of sending a large number of DNS requests asking for names that look like a random set of characters can be used to detect this type of activity.

It is unusual that a real DNS name would be random, because they are created to be easily remembered by humans. Statistical information can be extracted from the domain names requested, and a degree of randomness in them can be detected. If the randomness can be determined before the ransomware resolves a valid name, then it can be blocked

TABLE 3. Ransomware detection tools and input information used.

Ref	Date	Local dynamic												Local Static	Network				
		Data access information					Metadata access information				Function calls				DNS	DGA	SMB Traffic	General traffic	
		Entropy	File differences	Magic bytes	File removed	File extension	Canary files	Read/Write frequency	Amount of files	Dirs	Functions	Frequency	Cryptographic primitives						
M. M. Ahmadian et al. [89]	09/2015																		
Paik et al. [63]	05/2016				Yes			Yes	Yes								Yes		
N. Scaife et al. [59]	06/2016	Yes	Yes	Yes				Yes											
K. Cabaj et al. [88]	07/2016															Yes			
C. Moore [53]	08/2016						Yes												
D. Sgandurra et al. [50]	09/2016					Yes		Yes		Yes	Yes	Yes							
M. M. Ahmadian et al. [71]	09/2016					Yes			Yes	Yes		Yes							
F. Mbol et al. [51]	10/2016	Yes	Yes																
M. Shukla et al. [60]	10/2016	Yes						Yes	Yes	Yes									
A. Continella et al. [70]	12/2016	Yes						Yes				Yes							
Y. Feng et al. [65]	05/2017																Yes		
S. Chadha et al. [77]	05/2017																Yes		
E. Kirda [62]	08/2017	Yes			Yes			Yes											
R. Vinayakumar et al. [55]	09/2017										Yes	Yes							
Z. Chen et al. [58]	09/2017										Yes	Yes							
A. Kharraz et al. [61]	10/2017	Yes			Yes	Yes		Yes		Yes									
T. Lu et al. [66]	12/2017							Yes	Yes	Yes	Yes	Yes	Yes						Yes
M. M. Hasan et al. [67]	12/2017							Yes			Yes	Yes			Yes	Yes			Yes
JA. GÁmez-Hernández et al. [85]	12/2017							Yes											
S. S. Khashif et al. [49]	01/2018	Yes					Yes	Yes			Yes		Yes		Yes				
M. Aliaam et al. [56]	02/2018										Yes	Yes							
F. Quinkert et al. [97]	03/2018															Yes			
B. A. S Al-rimy et al. [57]	07/2018										Yes	Yes	Yes						
H. Zhang et al. [68]	08/2018							Yes			Yes	Yes	Yes						
R. Moussaileb et al. [54]	08/2018						Yes		Yes										
S. Mehnaz et al. [72]	10/2018	Yes	Yes			Yes	Yes	Yes			Yes		Yes						
D. Morato et al. [73]	12/2018																	Yes	
A. O. Almarshadani et al. [39]	03/2019															Yes	Yes	Yes	Yes
K. Lee et al. [52]	07/2019	Yes																	
Android tools																			
T. Yang et al. [69]	05/2015								Yes	Yes					Yes				
N. Andronio et al. [74]	12/2015														Yes				
S. Song et al. [64]	03/2016				Yes			Yes											
F. Mercado et al. [98]	06/2016															Yes			
M. Kanwal et al. [83]	05/2017															Yes			
A. Karimi et al. [79]	10/2017															Yes			
J. Chen et al. [75]	12/2017	Yes									Yes								
D. Maiorca et al. [78]	06/2018														Yes				

before it starts removing files. This can be a useful technique in a zero-day scenario because it does not require extracting information from a known ransomware. However, it is only successful against strains that use a DGA.

### 3) GENERAL TRAFFIC

A ransomware detection algorithm can use statistics extracted from network traffic. There are tools that take into account the number of different IP addresses accessed, the number of different server ports, and the number of sessions from different application level protocols. These parameters can enhance the decisions taken by a ML algorithm; however, they are never used as the sole input statistic.

In corporate environments, document storage is accomplished using shared network volumes. Most ransomware strains are capable of encrypting files stored on the network-mounted volumes in the infected hosts. The network traffic created by reading, writing, and destroying files can be used for ransomware detection.

## IV. RANSOMWARE DETECTION ALGORITHMS

The ransomware detection algorithms proposed in the literature use one or more of the parameters described in Section III, and decide whether to classify the software as benign or malign. In this section, we describe the different proposals on how to reach this decision. We provide a brief description of each algorithm and its target environment.

In Section VI we discuss the results obtained from each proposal in terms of successful detection, false detection, and resources required. We also discuss the limitations of each technique.

We grouped the algorithms based on the generic characteristics of their design. In Section IV-A we describe algorithms based on the combination of different input parameters. These algorithms provide a numeric risk value which, based on a threshold, determines whether software actions are classified as ransomware. This procedure requires tuning the mathematical combination of the input parameters and establishing the threshold. In Section IV-B we describe the proposals based on the application of ML algorithms to different sets of statistics extracted from ransomware action. ML techniques are usually supervised and therefore require a previous training phase. Finally, Section IV-C groups the proposals based on specific ransomware behaviour without the combination of numerical metrics. This last group presents ad-hoc detection procedures.

Table 3 shows the references from the scientific literature, the input parameters each one requires, and the type of detection algorithm based on the classification presented in this section.

### A. COMBINATIONAL TECHNIQUES

Most parameters described in Section 3 alone can result in the erroneous judgement of benign software as ransomware.

For example, a process that encrypts files can be legitimate, but the function calls it uses may result in its classification as ransomware. A program that accesses many files and removes them like malware may be a harmless compression utility. A program that overwrites a large number of files may be a photo edition tool applying a filter to a set of images. However, a program that accesses many files spread throughout many different directories and removing them and creating new files with random-looking content is less likely to be benign software (although it still may be). The more actions taken into account and the more of them present in malware activity, the higher the successful identification rate. However, collecting all these actions requires giving permission to the malware to run freely for a long time, long enough to encrypt and destroy several files.

The ransomware detection proposals presented in this section are a combination of a subset of parameters from those described in Section III and Figure 4. These parameters provide a numerical measure of software dynamic activity. These measurements are combined to create a single value that characterises process activity. Based on predetermined thresholds for this combined measurement or for each value, the process can be labelled as benign or ransomware.

Scaife et al. described in [59] software that measured disk access actions from each process while it was running. The analysis software grouped its input measurement parameters into primary and secondary indicators, being the three primary indicators more reliable for malware identification. All primary indicators must occur for the process to be classified as ransomware. As primary indicators, it used the identification of changes in file format (based on the “file” utility and the “magic bytes”), a quantitative comparison of the original and modified files, and an entropy measurement of the created files. The secondary indicators were the number of files deleted and the diversity of types in files read by the process. The risk evaluation for a process was incremented with time as a process performs actions in some of these parameters classified as dangerous. When this risk value reached a certain threshold, an alert was triggered and disk access was blocked for the process. However, there was no description of the determination of the threshold values.

Paik et al. [63] investigated the special behaviour of Solid State Disks (SSDs) where the data were not deleted when overwritten; they were deleted later, on a garbage collection action. They suggested introducing in SSDs the measurement of read-and-rewrite and deletion requests, and comparing the ratio of these actions to a threshold to detect a ransomware attack. They did not provide details regarding the dynamic measurement, threshold computation, or evaluation of the achievable results.

The anti-ransomware software described by Kharraz and Kirida in [61] was based on at least half a dozen input parameters related to the disk access actions performed by a process. From a file content perspective, it measured the variation in entropy from a read to a write request in the same section of a file; it also monitored the proportion of file content

overwritten and whether the process deleted files. The solution presented also collected metadata information regarding disk access actions. It checked whether a process accessed a large number of files with write privileges and whether the accessed files were from very different file types (from very different applications) or from the same one. In addition, it monitored the time between write requests and assigned a larger risk value the shorter the time. All these measured process features were combined to offer a risk assessment using a linear function. The weights in this function were determined by recursive feature elimination.

Moore [53] set a honeypot where file changes were monitored and different levels of alarm were triggered based on the threshold reached. Parameter and threshold configuration was obtained from the average normal activity over a day; however, no description was provided as to the determination of normal activity. The threshold values simply doubled normal average values. No validation against real ransomware was provided.

The method presented by Mbol et al. [51] restricted vigilance to image files (JPEG format). It measured the number of image files opened and the difference between the read and written versions. JPEG format contains compression characteristics, making it difficult to differentiate from an encrypted file using, for example, the entropy value of file content. The proposal used the Kullback-Lieber divergence to compare an original file and a possibly encrypted file. The thresholds for ransomware detection were decided by trial and error using TorrentLocker as the ransomware under analysis.

In [73], we measured network traffic between an infected computer and network shared volumes by measuring the number of bytes read and written as well as file content destroyed. These values are combined and checked against thresholds precomputed to achieve 100% identification for the available ransomware samples and minimum false positive events in the case of benign activity in a real scenario.

The previous proposals targeted the Microsoft Windows environment, while Song et al. [64], Andronio et al. [74], and Chen et al. [75] focused on ransomware detection in an Android cell phone operating system. The HelDroid proposal of Andronio et al. [74] combined the detection of threatening text in the binary or network traffic with a static analysis of the software searching for file access followed by encryption and deletion. This was combined with heuristics attempting to detect procedures to lock access to the device. In [64], Song et al. suggested monitoring any activity on a set of protected files, warning the user when a non-authorized process attempted to access any of those files. It also monitored CPU, memory, and disk usage, warning the user when a threshold was exceeded. No information was provided regarding how the time-varying statistical measurements should be taken or the thresholds computed. The proposal from Chen et al. [75] differed substantially. It was based on the monitoring of written files and detecting whether they were encrypted using the computation of file content entropy. Once a new encryption process was detected, the user interface was analysed to

determine whether it was the result of user actions or unknown to the user and therefore the result of encrypting ransomware.

### B. MACHINE LEARNING-BASED PROPOSALS

ML techniques differ from combinational methods because they typically use a much larger number of input parameters, a complex combination of them, and automatic training procedures for computing the weights in the combination. The ransomware detection studies based on ML have used mainly supervised learning techniques; however, some unsupervised proposals exist.

Continella et al. developed in [70] Microsoft Windows kernel modules which monitored process file system actions such as folder listing; files read, renamed, or written; and file entropy in write operations. They collected disk access data from benign applications running on computers from 11 volunteers and ransomware activity data from samples running in a controlled virtualised environment. They observed the data in different time scales to detect ransomware based on short-term and long-term behavioural patterns. The classifier for each scale was the random forest supervised learning algorithm.

The EldeRan proposal from [50] was based on the application of a supervised regularised logistic regression ML algorithm to a large set of features obtained from static and dynamic analysis of a suspect program. The training phase occurred offline after monitoring goodware and malware software in a sandbox environment. More than 30,000 features were extracted related to Windows API calls, registry key operations, file and directory operations, newly created (dropped) files, and strings present in the binary. Using feature selection based on a mutual information criterion [76], the set was reduced to 400 features while maintaining the best results.

In [67], Hasan and Rahman used a Support Vector Machine (SVM) classifier as the machine learning algorithm. More than sixty thousand features were extracted from static and dynamic operations (API calls, registry operations, file system operations, process operations, network actions). Similar to [50], a mutual information criterion was used to reduce the number of features to around 600. In [58] Chen et al. obtained more than 70,000 features from the flow graph of API calls used by the suspect process. They applied correlation techniques to reduce the dimensionality and compared the results using a Random Forest, a Support Vector Machine, Simple Logistic algorithm and the Naive Bayes. Meanwhile, 2entFOX from [71] used twenty different features, from cryptographic API calls to Windows registry modification, to design a Bayesian network.

Lu et al. [66] used the V-detector; a negative selection algorithm for classification. This supervised learning algorithm was inspired by the behavior of biological immune systems. The authors used features extracted from the API functions employed, network operations or memory patterns from the ransomware process. The number of features was reduced

using a low variance filter, which eliminated features with variances below a given threshold.

Vinayakumar et al. [55] used the Windows API calls from the process and their frequencies as the sole input features for the ML classifier. Both an SVM classifier and a multi-layer perceptron were trained. Chadha and Kumar [77] compared the classification results from several supervised and unsupervised algorithms, however, the only feature used as input was the set of domain names generated, trying to detect families of ransomware which used a Domain Generation Algorithm.

The proposal from Alam et al. [56] was based on the observation of the hardware performance counters present in modern processors. These are special registers that account for the frequency of different machine-level operations taken by the CPU. The goal was to detect high encryption activities based on the CPU events that occurred for these operations. When no ransomware was running, the sampled values from these counters were used for an autoencoder, an artificial neural network used in an unsupervised learning algorithm. No data from ransomware examples were provided in the training phase and their detection was based on detecting an anomaly that deviated from learned behavior.

Moussaileb et al. [54] analysed the file system traversal action performed by the ransomware to locate files to encrypt. They suggested using decoy (canary) directories, i.e. directories that should not be accessed by benign processes but are a target for ransomware. They compared the classification results from a Random Forest, a decision tree and the k-nearest neighbours algorithm.

There have been proposals which have not included every measured feature as an input to the ML algorithm but were designed with a hybrid approach of ML detection, threshold based detection, and ad-hoc mechanisms. For example, Mehnaz et al. [72] used an ML algorithm for the classification based on file access primitives (rapidly opened, read and written files). They complemented the detection system with the use of canary files and monitored file encryption as an increase in file-content entropy. They selected a Random Forest classifier after testing other options like Decision Trees, Naive Bayes or Logistic Regression.

Proposals like RansomWall [49] resorted to a machine learning classification only when a certain number of features were detected in the suspect process. The authors inspected the program binary before execution, checking the use of certain API calls and text strings. They also tracked the access and modifications to canary files and directories. When a certain number of features were present, they used an ML algorithm for the final process classification. They tested the results using supervised algorithms like Logistic Regression, Support Vector Machines, Artificial Neural Networks, Random Forests and Gradient Tree Boosting.

Almashhadani et al. [39] described a detection system that extracted the features exclusively from network traffic. The authors obtained twenty features from the characteristics of TCP, HTTP and DNS traffic. They built a decision making

module based on two classifiers. One classifier was based on per-packet features, while the second one was based on flow-based features. It did not require that both classifiers detected the ransomware. The classifiers evaluated were Random Forests, Bayes Networks, Support Vector Machines and Random Trees. The algorithm that provided the highest accuracy was selected for each classifier.

Machine learning algorithms have also been applied to crypto-ransomware detection in the Android operating system. R-PackDroid [78] counts the number of uses of different system API packages in the binary program. Based on these data, a Random Forest is used to classify the samples into three possible classes: benign, ransomware or generic malware. Karimi and Moattar's [79] ransomware detection method was also based on the analysis of an Android binary. It calculated the proportion of occurrences of each possible operation in the Android Dalvik Opcode language. They used LDA (Latent Dirichlet Allocation) as an image reduction and classification technique.

Lee et al. [52] used ML techniques to detect encrypted files before they were copied to a backup filesystem. In their proposal, the training phase was implemented at the backup system. It classified files from different users and file types and decided threshold values for file entropy. These thresholds were sent to the client hosts to decide whether a new version of the file was encrypted or not. They used three different methods of entropy computation as input data to the ML algorithm and compared the results from Decision Trees, k-Nearest Neighbours, Kernel Support Vector Machines and the Multi-Layer Perceptron.

Some commercial security and backup software solutions have introduced machine learning algorithms into their process. RansomFlare [80] and Acronis True Image [81] are examples we have found. Although the developers offered little information regarding the tools, in the case of [80], we know they were based on the actions performed by the analysed software and on SVM for the classification task.

### C. AD-HOC PROPOSALS

The proposals of the studies in this section used the knowledge of specific behaviour in ransomware samples and are valid only for a few ransomware families. In some cases, they measured sufficient software characteristics to take into account different behaviours.

Static analysis of a binary in search of specific patterns is the most usual form of malware fingerprinting. In the Android operating system, Cimitile et al. [82] offered a new approach, transforming Java bytecode into formal models and constructing temporal logic properties to describe the characteristic behaviour of a ransomware. The solutions of Kanwal et al. [83] were based on searching for specific static text strings in the binary, which can be indicative of a ransom request screen.

Canary files, decoy files, or honeyfiles have been used in several of the above-mentioned proposals [49], [53], [54],

[65], [72], [84], [85]. In [85], canary files were the only detection technique proposed.

Commercial ransomware detection tools such as CryptoStopper [86] are based on canary files, while others [87] monitor the extensions of newly created files in search of those used by known ransomware variants.

Studies by Cabaj and Mazurczyk [88] or by Ahmadian et al. [89] were based on detecting and blocking the communication to the ransomware C&C servers. Cabaj and Mazurczyk [88] used OpenFlow-capable switches to intercept or monitor DNS requests which were checked against a blacklist of known C&C domain names. The proposal of Ahmadian et al. [89] accommodated ransomware families which generated pseudo-random domain names. It attempted to detect DNS requests to names generated by a DGA based on a Markov chain and a model for the frequency of alphabetic characters and character transitions. Commercial offerings such as Cisco Umbrella [90] are also based on DNS inspection and domain blacklisting.

## V. FILE RECOVERY

We have described ransomware detection techniques capable of detecting malware before it starts encrypting files and some other proposals that require witnessing ransomware action to raise an alarm. In cases where the ransomware has already encrypted some files before it is halted, there is some loss of user data. In this section we describe techniques which avoid any loss of data because they are capable of restoring the original file content.

The desired outcome of file recovery is the last version of file content before the encryption. Most backup policies only recover the last backup version file, which is not necessarily the last version the user edited. Some proposals described in this section are capable of recovering the last edited version.

### A. CRYPTOGRAPHIC RECOVERY

Files encrypted by ransomware can be recovered if the decryption key can be obtained (obviously without paying the ransom). Depending on the key management procedure implemented or the encrypting functions used, this task can be accomplished by monitoring the malware action.

For certain ransomware families, the decryption keys are available to the public. They were extracted from the binary or the ransomware used weak encryption procedures vulnerable to cryptanalysis techniques [91], [92]. The No More Ransom Project webpage [93] offers links to decryption tools for at least 100 ransomware versions.

In PayBreak [94], the authors took advantage of the fact that hybrid cryptosystems create encryption keys for a symmetric encryption algorithm that are used in file encryption, although they are later encrypted using a public key whose private key is only known by the attacker. Being present in the infected computer, at least during file encryption, these keys can theoretically be retrieved. The authors of [94] developed a software component that intercepted the API calls to

session key creation functions and stored the returned keys in a protected key vault. Using these keys, the encrypted files were recovered. They hooked into system-provided encryption libraries and attempted to recognise statically linked libraries using fuzzy signatures.

A similar proposal to Paybreak [94] was presented by Kim et al. [95], where the intercepted function was used for random number generation in the process of session key creation. The random numbers provided were controlled by the ransomware protection software, and by using the same numbers, it recreated the session keys for file decryption.

## B. RECOVERY FROM BACKUPS

Without an effective ransomware detection mechanism, most enterprises resort to recovery from backups when they suffer a ransomware attack. These backups usually do not contain the last version of files; therefore, some data are always lost. They also suffer the effect of work disruption during file recovery.

Using the special writing behaviour in an SSD, a better backup policy can be implemented. In FlashGuard [96], Huang et al. took advantage of how SSDs perform out-of-place write operations instead of replacing the old data with the new content of a file. SSDs behave in this manner because they must erase the data content before they write to those memory cells, and the erase operation is time consuming. Since the SSD does not erase the original data but writes to a different memory cell, these data can be recovered if they are read before they are erased by the SSD garbage collector. However, this procedure does not offer the detection of ransomware action. Paik et al. [63] used a very similar approach, but added ransomware detection based on the read/rewrite patterns.

Some proposals for ransomware detection require observing the process of file encryption, and raise an alarm only when some files have been lost. This procedure can be improved by keeping a backup of the files being edited at the cost of a higher disk response time to applications because any file modification operation is intercepted to maintain a backup.

Continella et al. [70] created a copy of a file before allowing the requesting process to modify it. If the process was classified as ransomware, it was halted and the files it modified were recovered from the copied documents, which were the most recent versions before encryption. Following a similar procedure, in [61] the monitoring software redirected every write request to a sparse file in a protected area without changing the original file. In [60] the read/write operations were redirected only in cases of suspicious file system activity.

Storage systems based on objects instead of files can make objects immutable, turning write operations into the creation of a new version of the object. Eze et al. described in [18] the use of secure object storage to recover encrypted objects.

Both Baykara and Sekin [99] and Jin et al. [100] described backup solutions against ransomware attacks. They created

periodic copies of protected files in either a compressed file [99] or inside a docker container [100], adding the possibility of an extra backup to a remote file server. A similar proposal was presented in [101] for an Android operating system. These proposals are similar to some recent commercial offerings of backup solutions targeting ransomware threats. Most solutions offer no significant difference to a usual backup and only expose how a backup policy can solve or alleviate the problems that result from a ransomware infection [102]. However, products like [80], [81], [103] combine the detection of ransomware action with the recovery mechanisms of backup files.

## VI. DETECTION AND PERFORMANCE RESULTS

All the proposals found in the literature presented a tradeoff between detection results and performance. Table 4 shows a summary of the main results extracted from these studies. In this section, we explain the content of each column in Table 4 and highlight the main differences encountered. Empty table cells are the result of a parameter that could have been measured but for which no relevant data were found in the article.

### A. DETECTION RATE

The first column of Table 4 shows the percentage of ransomware samples detected by a system. This is the most prevalent result presented, and most studies offered detection results above 90%. In most cases, detection performance was defined as the quotient between the number of samples detected and the total number of ransomware samples, i.e. the probability of detecting a sample. However, Zhang et al. [68] defined detection performance as the quotient between the number of samples detected and the sum of true and false detections, i.e. the probability that an alarm was correct. Karimi and Moattar [79] provided the proportion of samples (ransomware or benign) that were correctly classified as the closest metric. There is no homogeneous definition for this performance result.

The detection rate alone does not provide a fair comparison because of the different populations of ransomware samples employed. The column 'number of samples' in Table 4 details the number of ransomware samples and/or ransomware families that were used in the tests or in the learning phase. Proposals such as Proposals such as 2entFOX [71] or R-Locker [85] were capable of detecting all the offered samples but were tested using only a few ransomware families. This was also the case in [88], [97]. In a few studies, a single ransomware family was tested [56], [65], making the results hardly extensible. An extreme case is [51], targeting only JPEG files being encrypted by TorrentLocker. Lee et al. [52] used encrypted files in the ML training and test phases but the authors did not specify how those files were encrypted. In these last two cases the detection rate was not measured using ransomware samples but encrypted files, so it is not comparable to other articles. In most proposals, although only a few ransomware families were detected, the methodology could be valid for

**TABLE 4. Performance and detection results presented in the literature.**

Ref	Detection Rate	False positives	Detection time	Lost files	Impact on resources	Ransomware samples (Families)	Benign applications
M. M. Ahmadian <i>et al.</i> [89]	50%	0		0		20 (14)	
Paik <i>et al.</i> [63]							
N. Scaife <i>et al.</i> [59]	100%	1 app (7zip)		Mean: 10 Maximum: 30	9-16ms delay	492 (14)	30
K. Cabaj <i>et al.</i> [88]				0		332 (CryptoWall)	
C. Moore [53]	100%					ransomware simulator	
D. Sgandurra <i>et al.</i> [50]	93.3%	1.6%				582 (11)	942
M. M. Ahmadian <i>et al.</i> [71]	100%	0%				8 (8)	8
F. Mbol <i>et al.</i> [51]	>90%	0.05%-0.25%		All		1 (1)	4,000 files
M. Shukla <i>et al.</i> [60]				<20	1-2% CPU		50 users working
A. Continella <i>et al.</i> [70]	97.7%	0.038%		0 (recovered)	runtime overhead 0.26	688 (18)	2,245 (11 users)
Y. Feng <i>et al.</i> [65]	100%		"tolerable" time	0	"low"	1(1)	
S. Chadha <i>et al.</i> [77]	>85%	<10%	110ms	0		131 (domains)	3473 domains
E. Kirda [62]	96.3%	0%				2,121	149
R. Vinayakumar <i>et al.</i> [55]	98%	<1%				755 (7)	219
Z. Chen <i>et al.</i> [58]	97.6%	1.2%				83	85
A. Kharraz <i>et al.</i> [61]	100%	0.8%		0 (recovered)	7-9% overhead	1,181 (29)	230 GB (5 users)
T. Lu <i>et al.</i> [66]	95%	7-8%				1,000	1,000
M. M. Hasan <i>et al.</i> [67]	97%	3%				360 (20)	460
JA. Gómez-Hernández <i>et al.</i> [85]	100%					3 (3)	
S. S. Khashif <i>et al.</i> [49]	98.25%	0.56%		0	<2%	572 (12)	442
M. Alaam <i>et al.</i> [56]	100%	"almost" 0%	3.43 seconds	n		1 (1)	
F. Quinkert <i>et al.</i> [97]	96%			0		1 (1)	
B. A. S Al-rimy <i>et al.</i> [57]	>90%					8,152 (5)	1,000
H. Zhang <i>et al.</i> [68]	99.8%		4.26 seconds			1,787 (8)	100
R. Moussaileb <i>et al.</i> [54]	99.35%	<1%				694 (9)	617
S. Mehnaz <i>et al.</i> [72]	100%	0.9%	3.45 seconds	<10	Memory: 70 MB Overhead: 1.9% Delay: 10ms	14 (14)	261 processes
D. Morato <i>et al.</i> [73]	100%	1 in 10 billion opened files	less than 20 seconds	10 (recovered)	"No impact"	54 (19)	4TB (traces of hundreds users)
A. O. Almashhadani <i>et al.</i> [39]	97%	2%				4 (1)	10 PCAPs (3.5 GB)
K. Lee <i>et al.</i> [52]	100%	0%				Unknown	1200 files
<b>Android tools</b>							
T. Yang <i>et al.</i> [69]							
N. Andronio <i>et al.</i> [74]						650	81046
S. Song <i>et al.</i> [64]	100%						
F. Mercaldo <i>et al.</i> [98]	99.56%	0%				683	600
M. Kanwal <i>et al.</i> [83]	100%						
A. Karimi <i>et al.</i> [79]	97.5%					250 (3)	30
J. Chen <i>et al.</i> [75]	99%	0.09%	5 seconds	Recovered	5% memory	2,721	9,238
D. Maiorca <i>et al.</i> [78]	97%	1%				2,047	4,098

other samples [79], [89] because it was not locked to a specific behaviour in a ransomware family.

Testing a detection algorithm with a wide variety of ransomware families is problematic. Ransomware is volatile in the sense that most samples become inactive after a few months because of C&C servers being taken down. Therefore, testing with a large sample test is not always possible, and some proposals enjoy the simple advantage of being developed when many new active samples are discovered. In addition, the older a study, the less families it can be tested against, which hinders the comparison of results.

We observed a wide range of sample sizes used in different studies, from tens or hundreds [50], [59], [61], [67], [70], [89] to thousands [57], [68], [75], [78]. The studies using thousands of samples resorted to automatic download from antivirus repositories. It is difficult to evaluate how different these samples are and whether they represent a more diverse population of ransomware behaviour than the set used by a study with only tens of samples. Although the hashes of the binaries were all different, there may be only a small set of different behaviours for the evaluation of a detection algorithm.

### B. FALSE POSITIVES

The second column of Table 4 shows the probability of the detection algorithm classifying benign software as ransomware. In most cases, this is a lesser mistake compared to allowing malign software to run. However, a large number of false positives means that a user's work will be disrupted more often, which in turn may lead the user to disable the detection software.

Not all studies offered a measurement of the percentage of false positives, and those that did showed results in the single-digits. However, the validity of the figures in this column depends on the population of benign software behaviour that was tested, or the population of users running the software because of the dependency on user actions for some detection algorithms. The column "number of user applications" offers a summary of the data provided by the authors regarding the tests that lead to false positive results. It was not always based on the number of applications; it was based on the number of DNS requests when the detection algorithm was based on the traffic from this protocol.

Similar to the detection rate, the results were not easily comparable. A few authors offered their data to encourage reproducibility; however, no uniform testing data were found and would be feasible only when the input information to the detection algorithms were the same.

### C. CPU IMPACT

Most proposals were based on software running on a user's host computer. Therefore, they required CPU and hard disk resources. The exception was any detection solution based on network traffic alone which could be deployed in firewalls, network probes, or traffic inspection devices.

The column "resources required" provides the data on CPU cycle consumption published by the authors. Sometimes it was a percentage of CPU load, an increased delay in response time, or only a qualitative result (e.g. "low"). Not many authors evaluated the effect of their detection approach on computer responsiveness. However, it can be argued that the presented implementations were research prototypes open to performance improvement, and therefore, the results would not be conclusive.

### D. LOST FILES

Finally, some proposals detected and stopped ransomware action before it encrypted files, while other detectors required monitoring ransomware action (the encryption phase) to label it as dangerous. The result was a time window when some files could be lost before the ransomware was detected.

The column "lost files" shows the data offered by the authors on this topic. Studies with a "0" in this column lost no files. In some cases, this was the result of detection prior to the encryption phase. In other cases, the monitoring software performed file backups [61], [63], [65], [70], [72] or intercepted the disk access system calls from any unclassified software.

Once intercepted, the system calls could be redirected to a copy of the original file.

### VII. OPEN ISSUES

The panorama of ransomware families has been changing in the last five years. They have increased their resistance to cryptanalysis techniques more than to detection techniques. Nonetheless, new strains have shown improved code and behaviour obfuscation. For example, CTBLocker [30] encrypts only part of a file, resulting in lower writing rates than previous ransomware; therefore, a detection based on disk activity may fail. GandCrab obfuscates strings used as function call parameters [24].

The best protection in the literature is offered by a combination of detection and backup policies where the original file can be recovered. Even if automatic detection fails, eventually user manual detection takes place and the user finds the files that were encrypted. The main drawback in these proposals is the cost in CPU and disk usage. The response time to read and write actions is increased because of backup maintenance. Backups also take up space in the local disk. In the case of detection failure, it is not likely that a local disk will be capable of keeping a copy of all the encrypted files because it will take up as much space in the backup as the original files. In some proposals, the backup takes place in a network volume which reduces local disk usage but increases the configuration burden, requires a disk server, and reduces the frequency of backups.

There is a lack of evaluation of the different proposals, especially in their usability because of the added load to the CPU and disk. Most proposals are proof-of-concept developments from academia and lack the optimisations needed to reach conclusions regarding usability. The proposed algorithms can be effective but not efficient and incur grave performance degradation. There is a need for better evaluation of the effects these detection algorithms have on computer responsiveness. Proposals deployed in network equipment or network traffic probes suffer less stringent requirements; however, they have a more limited visibility of ransomware action than locally installed analysis software.

As shown in Section VI, the results offered in the literature for the different proposals were difficult if not impossible to compare. They did not target the same ransomware families and presented results using different metrics. A unified evaluation and comparison scenario is needed for a serious reproducible research. As a first step, the list of malware binaries should be available for comparison. This is a very difficult condition because ransomware stops working if the required C&C servers are taken down. As a second step, a set of unified metrics for evaluation is required. The way of determining true and false positives and true and false negatives is not the same. Finally, almost no proposal provided a source code or a sufficient description of the algorithm to reproduce its behaviour. Algorithms based on thresholds do not provide the exact procedure to measure the input information and the exact thresholds that produce the presented values.



Research results based on automatic learning algorithms do not offer the output of the training, which can lead to a comparison of the behaviour with new malware samples. Overall, nowadays, comparability and reproducibility is neither facilitated by the problem at hand nor by the way researchers present their results.

## VIII. CONCLUSION

This survey focused on the proposed algorithms for the detection of cryptographic ransomware. We analysed 63 ransomware samples from 52 families to extract the characteristic steps taken by a ransomware. From this knowledge, we detailed the input data that detection algorithms extract from ransomware based on each of these steps. We classified these input data based on the point of extraction (local to the infected machine or remotely) and whether it was extracted before the malware ran or during its activity. The algorithms were grouped based on this classification.

We presented a second classification of ransomware detection algorithms based not only on the input parameters used but also on how they combine these data to reach a classification result of either benign or malign software. In the last years, algorithms based on ML techniques have become very popular and have been used in more than 35% of the studies in the literature. However, they coexisted with a large set of heuristics and ad-hoc detection solutions.

After exploring the input data and the procedure in the detection algorithms, we collected the evaluation and comparison results presented in the literature for each detection proposal. They were scarce and difficult to compare, leaving room for a more reproducible research in the field of security.

Finally, we discussed the open issues to offer effective solutions for ransomware detection or to support better research and validation of the proposed algorithms.

## REFERENCES

- [1] "Internet organised crime threat assessment (IOCTA) 2018," EUROPOL, The Hague, The Netherlands, Tech. Rep. QL-AL-18-001-EN-N, 2018. doi: [10.2813/858843](https://doi.org/10.2813/858843).
- [2] (Jun. 2019). *Ransomware Repercussions: Baltimore County Sewer Charges, 2 Medical Services Temporarily Suspended*. Accessed: Jul. 4, 2019. [Online]. Available: <https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/ransomware-repercussions-baltimore-county-sewer-charges-2-medical-services-temporarily-suspended>
- [3] S. Cobb. (Dec. 2018). *Ransomware vs. Printing Press? US Newspapers Face 'Foreign Cyberattack'*. Accessed: Jul. 4, 2019. [Online]. Available: <https://www.welivesecurity.com/2018/12/31/ransomware-printing-press-newspapers/>
- [4] "Internet organised crime threat assessment (IOCTA) 2014," EUROPOL, The Hague, The Netherlands, Tech. Rep. QL-AL-14-001-EN-E, 2014. [Online]. Available: <https://doi.org/10.2813/16>
- [5] D. Su, J. Liu, X. Wang, and W. Wang, "Detecting Android locker ransomware on chinese social networks," *IEEE Access*, vol. 7, pp. 20381–20393, 2019.
- [6] (Apr. 2019). *Coverware's Q1 2019 Global Ransomware Marketplace Report*. Accessed: Jul. 4, 2019. [Online]. Available: <https://www.coverware.com/blog/2019/4/15/ransom-amounts-rise-90-in-q1-as-ryuk-ransomware-increases>
- [7] (Oct. 2018). *Securing Cyber Resilience in Health and Care: Progress Update October 2018*. Jul. 4, 2019. [Online]. Available: [https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment\\_data/file/747464/securing-cyber-resilience-in-health-and-care-september-2018-update.pdf](https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/747464/securing-cyber-resilience-in-health-and-care-september-2018-update.pdf)
- [8] A. Benfield. (May 2017). *Cyber Event Briefing: WannaCry Ransomware Attack*. Accessed: Jul. 4, 2019. [Online]. Available: <http://thoughtleadership.aonbenfield.com/Documents/20170518-ab-cyber-wannacry-briefing.pdf>
- [9] R. Unuchek, F. Sinityn, D. Parinov, and V. Stolyarov, "IT threat evolution Q1 2017. Statistics," Kaspersky, Moscow, Russia, Tech. Rep., 2017. [Online]. Available: <https://securelist.com/it-threat-evolution-q1-2017-statistics/78475/>
- [10] R. Unuchek. (Mar. 2018). *Mobile Malware Evolution 2017*. Accessed: Jul. 4, 2019. [Online]. Available: <https://securelist.com/mobile-malware-review-2017/84139/>
- [11] V. Chebyshev, F. Sinityn, D. Parinov, A. Liskin, and O. Kupreev, "IT threat evolution Q1 2018. Statistics," Kaspersky, Moscow, Russia, Tech. Rep., 2018. [Online]. Available: <https://securelist.com/it-threat-evolution-q1-2018-statistics/85541/>
- [12] P. Zavarsky and D. Lindskog, "Experimental analysis of ransomware on Windows and Android platforms: Evolution and characterization," *Proc. Comput. Sci.*, vol. 94, pp. 465–472, 2016.
- [13] S. Aurangzeb, M. Aleem, M. A. Iqbal, and M. A. Islam, "Ransomware: A survey and trends," *J. Inf. Assurance Secur.*, vol. 6, no. 2, pp. 48–58, 2017.
- [14] K. A. Gandhi, "Survey on ransomware: A new era of cyber attack," *Int. J. Comput. Appl.*, vol. 168, no. 3, pp. 38–41, 2017.
- [15] J. P. Tailor and A. D. Patel, "A comprehensive survey: Ransomware attacks prevention, monitoring and damage control," *Int. J. Res. Sci. Innov.*, vol. 4, pp. 2321–2705, 2017.
- [16] D. Gonzalez and T. Hayajneh, "Detection and prevention of crypto-ransomware," in *Proc. IEEE 8th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, Oct. 2017, pp. 472–478.
- [17] S. I. Popoola, B. U. Iyekepolo, S. O. Ojewande, F. O. Sweetwilliams, S. N. John, and A. A. Atayero, "Ransomware: Current trend, challenges, and research directions," in *Proc. World Congr. Eng. Comput. Sci.*, vol. II, 2017, pp. 1–6.
- [18] K. Eze, C. Akujuobi, M. N. O. Sadiku, and P. Chhetri, "An approach to changing ransomware threat landscape," *J. Sci. Eng. Res.*, vol. 5, pp. 68–74, Oct. 2018.
- [19] 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.
- [20] (Apr. 2019). *Ransomware*. Accessed: Jul. 4, 2019. [Online]. Available: <https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/ransomware-malware>
- [21] (May 2017). *WannaCrypt Ransomware Worm Targets Out-of-Date Systems*. Accessed: Jul. 4, 2019. [Online]. Available: <https://www.microsoft.com/security/blog/2017/05/12/wannacrypt-ransomware-worm-targets-out-of-date-systems>
- [22] (May 2017). *WannaCry Report*. Accessed: Jul. 4, 2019. [Online]. Available: [https://www.pandasecurity.com/mediacenter/src/uploads/2017/05/1705-Informe\\_WannaCry-v160-en.pdf](https://www.pandasecurity.com/mediacenter/src/uploads/2017/05/1705-Informe_WannaCry-v160-en.pdf)
- [23] (May 2016). *DMA Locker 4.0: Known Ransomware Preparing for a Massive Distribution*. Accessed: Jul. 4, 2019. [Online]. Available: <https://blog.malwarebytes.com/threat-analysis/2016/05/dma-locker-4-0-known-ransomware-preparing-for-a-massive-distribution/>
- [24] T. Boczan. (Jun. 2018). *The Evolution of GandCrab Ransomware*. Accessed: Jul. 4, 2019. [Online]. Available: <https://www.vmrays.com/cyber-security-blog/gandcrab-ransomware-evolution-analysis/>
- [25] (Oct. 2017). *Petya Ransomware Outbreak: Here's What You Need to Know*. Accessed: Jul. 4, 2019. [Online]. Available: <https://www.symantec.com/blogs/threat-intelligence/petya-ransomware-wiper>
- [26] S. Hansman and R. Hunt, "A taxonomy of network and computer attacks," *Comput. Secur.*, vol. 24, no. 1, pp. 31–43, 2005.
- [27] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. S. Gaur, M. Conti, and M. Rajarajan, "Android security: A survey of issues, malware penetration, and defenses," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 998–1022, 2nd Quart., 2015.
- [28] (May 2017). *Command and Control [C&C] Server*. Accessed: Jul. 4, 2019. [Online]. Available: <https://www.trendmicro.com/vinfo/us/security/definition/command-and-control-server>

- [29] (Mar. 2017). *Explained: Spora Ransomware*. Accessed: Jul. 4, 2019. [Online]. Available: <https://blog.malwarebytes.com/threat-analysis/2017/03/spora-ransomware/>
- [30] (Jan. 2017). *Know Your Ransomware: CTB-Locker*. Accessed: Jul. 4, 2019. [Online]. Available: <https://www.secalliance.com/blog/ransomware-ctb-locker/>
- [31] L. Constantin. (Jun. 2016). *Bart Ransomware Shows it Can be Effective Without Sophisticated Encryption*. Accessed: Jul. 4, 2019. [Online]. Available: <https://www.pcworld.com/article/3088581/bart-ransomware-shows-it-can-be-effective-without-sophisticated-encryption.html>
- [32] R. Lipovsky and P. Kálnai. (Jan. 2017). *KillDisk now targeting Linux: Demands \$250K Ransom, But Can't Decrypt*. Accessed: Jul. 4, 2019. [Online]. Available: <https://www.welivesecurity.com/2017/01/05/killdisk-now-targeting-linux-demands-250k-ransom-cant-decrypt/>
- [33] F. Bacurio, J. Salvio, and R. Joven. (Feb. 2017). *A Closer Look at Sage 2.0 Ransomware Along with Wise Mitigations*. Accessed: Jul. 4, 2019. [Online]. Available: <https://www.fortinet.com/blog/threat-research/a-closer-look-at-sage-2-0-ransomware-along-with-wise-mitigations.html>
- [34] M.-E. M. Léveillé. (Feb. 2017). *New Crypto-Ransomware Hits macOS*. Accessed: Jul. 4, 2019. [Online]. Available: <https://www.welivesecurity.com/2017/02/22/new-crypto-ransomware-hits-macos/>
- [35] (Aug. 2018). *Ryuk Ransomware: A Targeted Campaign Break-Down*. Accessed: Jul. 4, 2019. [Online]. Available: <https://research.checkpoint.com/ryuk-ransomware-targeted-campaign-break/>
- [36] V. Hioureas. (Jan. 2018). *Scarab Ransomware: New Variant Changes Tactics*. Accessed: Jul. 4, 2019. [Online]. Available: <https://blog.malwarebytes.com/threat-analysis/2018/01/scarab-ransomware-new-variant-changes-tactics/>
- [37] (Mar. 2019). *What You Need to Know About the LockerGoga Ransomware*. Accessed: Jul. 4, 2019. [Online]. Available: <https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/what-you-need-to-know-about-the-lockergoga-ransomware>
- [38] (Nov. 2017). *Diving Into Chimera Ransomware*. Accessed: Jul. 4, 2019. [Online]. Available: <https://reaqta.com/2015/11/diving-into-chimera-ransomware/>
- [39] A. O. Almashhadani, M. Kaiiali, S. Sezer, and P. O'Kane, "A multi-classifier network-based crypto ransomware detection system: A case study of Locky ransomware," *IEEE Access*, vol. 7, pp. 47053–47067, 2019.
- [40] M.-E. M. Leveille, *TorrentLocker: Ransomware a Country Near You*. Bratislava, Slovakia: ESET, 2014.
- [41] S. Chaudhary. (Jul. 2018). *Tesla-Crypt Ransomware Analysis*. Accessed: Jul. 4, 2019. [Online]. Available: <https://medium.com/@sauravchaudhary/tesla-crypt-ransomware-analysis-e9b1dc5be0f>
- [42] V. Jain and R. Chong. (May 2016). *Locky Gets Clever!* Accessed: Jul. 4, 2019. [Online]. Available: [https://www.fireeye.com/blog/threat-research/2016/05/locky\\_gets\\_clever.html](https://www.fireeye.com/blog/threat-research/2016/05/locky_gets_clever.html)
- [43] (Aug. 2016). *New Cerber Ransomware Variant Packs Improved Key Generation*. Accessed: Jul. 4, 2019. [Online]. Available: <https://www.securityweek.com/new-cerber-ransomware-variant-packs-improved-key-generation>
- [44] (Jun. 2016). *Bart Ransomware Help & Support*. Accessed: Jul. 4, 2019. [Online]. Available: <https://www.bleepingcomputer.com/forums/t/618157/bart-ransomware-help-support-bartzip-recoverytxt/page-2>
- [45] A. Ajjan and D. Palotay. (Apr. 2018). *BTCWare Ransomware*. Accessed: Jul. 4, 2019. [Online]. Available: <https://www.sophos.com/en-us/medialibrary/PDFs/factsheets/sophos-btcware-ransomware-wpna.pdf>
- [46] (Oct. 2017). *BadRabbit: A Closer Look at the New Version of Petya/NotPetya*. Accessed: Jul. 4, 2019. [Online]. Available: <https://blog.malwarebytes.com/threat-analysis/2017/10/badrabbit-closer-look-new-version-petyanotpetya/>
- [47] (Jul. 2017). *Cerber Ransomware Is Not on Vacation This Summer*. Accessed: Jul. 4, 2019. [Online]. Available: <https://www.acronis.com/en-us/blog/posts/cerber-ransomware-not-vacation-summer>
- [48] *Virlock Ransomware*. Accessed: Jul. 4, 2019. [Online]. Available: <https://www.knowbe4.com/virlock-ransomware>
- [49] S. K. Shaukat and V. J. Ribeiro, "RansomWall: A layered defense system against cryptographic ransomware attacks using machine learning," in *Proc. 10th Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2018, pp. 356–363.
- [50] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, "Automated dynamic analysis of ransomware: Benefits, limitations and use for detection," Sep. 2016, *arXiv:1609.03020*. [Online]. Available: <https://arxiv.org/abs/1609.03020>
- [51] F. Mbol, J.-M. Robert, and A. Sadighian, "An efficient approach to detect torrentlocker ransomware in computer systems," in *Proc. Int. Conf. Cryptol. Netw. Secur.* Milan, Italy: Springer, 2016, pp. 532–541.
- [52] K. Lee, S.-Y. Lee, and K. Yim, "Machine learning based file entropy analysis for ransomware detection in backup systems," *IEEE Access*, vol. 7, pp. 110205–110215, 2019.
- [53] C. Moore, "Detecting ransomware with honeypot techniques," in *Proc. Cybersecurity Cyberforensics Conf. (CCC)*, Aug. 2016, pp. 77–81.
- [54] R. Moussaileb, B. Bouget, A. Palisse, H. Le Boudier, N. Cuppens, and J.-L. Lanet, "Ransomware's early mitigation mechanisms," in *Proc. 13th Int. Conf. Availability, Rel. Secur.*, 2018, p. 2.
- [55] R. Vinayakumar, K. P. Soman, K. K. S. Velan, and S. Ganorkar, "Evaluating shallow and deep networks for ransomware detection and classification," in *Proc. Int. Conf. Adv. Comput., Commun. Inform. (ICACCI)*, Sep. 2017, pp. 259–265.
- [56] M. Alam, S. Bhattacharya, D. Mukhopadhyay, and A. Chattopadhyay, "RAPPER: Ransomware prevention via performance counters," Feb. 2018, *arXiv:1802.03909*. [Online]. Available: <https://arxiv.org/abs/1802.03909>
- [57] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, "Redundancy coefficient gradual up-weighting-based mutual information feature selection technique for crypto-ransomware early detection," Jul. 2018, *arXiv:1807.09574*. [Online]. Available: <https://arxiv.org/abs/1807.09574>
- [58] 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.*, 2017, pp. 196–201.
- [59] 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. doi: [10.1109/ICDCS.2016.46](https://doi.org/10.1109/ICDCS.2016.46).
- [60] M. Shukla, S. Mondal, and S. Lodha, "Poster: Locally virtualized environment for mitigating ransomware threat," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. CCS*, 2016, pp. 1784–1786. doi: [10.1145/2976749.2989051](https://doi.org/10.1145/2976749.2989051).
- [61] A. Kharraz and E. Kirda, "Redemption: Real-time protection against ransomware at end-hosts," in *Research in Attacks, Intrusions, and Defenses*. Atlanta, GA, USA: Springer, 2017, pp. 98–119. doi: [10.1007/978-3-319-66332-6\\_5](https://doi.org/10.1007/978-3-319-66332-6_5).
- [62] E. Kirda, "UNVEIL: A large-scale, automated approach to detecting ransomware (keynote)," in *Proc. IEEE 24th Int. Conf. Softw. Anal., Evol. Reeng. (SANER)*, Feb. 2017, p. 1. doi: [10.1109/saner.2017.7884603](https://doi.org/10.1109/saner.2017.7884603).
- [63] J.-Y. Paik, K. Shin, and E.-S. Cho, "Poster: Self-defensible storage devices based on flash memory against ransomware," in *Proc. IEEE Symp. Security Privacy*, May 2016, pp. 1–2.
- [64] 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.
- [65] Y. Feng, C. Liu, and B. Liu, "Poster: A new approach to detecting ransomware with deception," in *Proc. 38th IEEE Symp. Security Privacy*, 2017, pp. 1–2.
- [66] T. Lu, L. Zhang, S. Wang, and Q. Gong, "Ransomware detection based on V-detector negative selection algorithm," in *Proc. Int. Conf. Secur., Pattern Anal., Cybern. (SPAC)*, Dec. 2017, pp. 531–536.
- [67] M. M. Hasan and M. M. Rahman, "RansHunt: A support vector machines based ransomware analysis framework with integrated feature set," in *Proc. 20th Int. Conf. Comput. Inf. Technol. (ICCIT)*, Dec. 2017, pp. 1–7.
- [68] H. Zhang, X. Xiao, F. Mercedo, 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.
- [69] T. Yang, Y. Yang, K. Qian, D. C.-T. Lo, Y. Qian, and L. Tao, "Automated detection and analysis for Android ransomware," in *Proc. IEEE 17th Int. Conf. High Perform. Comput. Commun., IEEE 7th Int. Symp. CyberSpace Saf. Secur., IEEE 12th Int. Conf. Embedded Softw. Syst.*, Aug. 2015, pp. 1338–1343.
- [70] A. Continella, A. Guagnelli, G. Zingaro, G. De Pasquale, A. Barenghi, S. Zanero, and F. Maggi, "ShieldFS: A self-healing, ransomware-aware filesystem," in *Proc. 32nd Annu. Conf. Comput. Secur. Appl. ACSAC*, 2016, pp. 336–347. doi: [10.1145/2991079.2991110](https://doi.org/10.1145/2991079.2991110).

- [71] M. M. Ahmadian and H. R. Shahriari, "2entFOX: A framework for high survivable ransomsware detection," in *Proc. 13th Int. Iranian Soc. Cryptol. Conf. Inf. Secur. Cryptol. (ISCISC)*, Sep. 2016, pp. 79–84.
- [72] S. Mehnaz, A. Mudgerikar, and E. Bertino, "RWGuard: A real-time detection system against cryptographic ransomware," in *Proc. Int. Symp. Res. Attacks, Intrusions, Defenses*. Heraklion, Greece: Springer, 2018, pp. 114–136.
- [73] 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.
- [74] N. Andronio, S. Zanero, and F. Maggi, "HelDroid: Dissecting and detecting mobile ransomware," in *Proc. Int. Symp. Recent Adv. Intrusion Detection*. Kyoto, Japan: Springer, 2015, pp. 382–404.
- [75] 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.
- [76] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE Trans. Neural Netw.*, vol. 5, no. 4, pp. 537–550, Jul. 1994.
- [77] S. Chadha and U. Kumar, "Ransomware: Let's fight back!" in *Proc. Int. Conf. Comput., Commun. Automat. (ICCCA)*, 2017, pp. 925–930.
- [78] D. Maiorca, F. Mercaldo, G. Giacinto, C. A. Visaggio, and F. Martinelli, "R-PackDroid: API package-based characterization and detection of mobile ransomware," in *Proc. Symp. Appl. Comput.*, 2017, pp. 1718–1723.
- [79] A. Karimi and M. H. Moattar, "Android ransomware detection using reduced opcode sequence and image similarity," in *Proc. 7th Int. Conf. Comput. Knowl. Eng. (ICCKE)*, Oct. 2017, pp. 229–234.
- [80] "Defeat ransomware with Varonis and NetApp," Varonis, New York, NY, USA, White Paper, 2016. [Online]. Available: <https://labs.mwrinfosecurity.com/assets/resourceFiles/mwri-behavioural-ransomware-detection-2017-04-5.pdf>
- [81] (2018). *Acronis True Image 2018 for PC*. Accessed: Jun. 2018. [Online]. Available: <https://www.acronis.com/es-es/support/documentation/ATIMAC2018/#39918.html>
- [82] A. Cimitile, F. Mercaldo, V. Nardone, A. Santone, and C. A. Visaggio, "Talos: No more ransomware victims with formal methods," *Int. J. Inf. Secur.*, vol. 17, no. 6, pp. 719–738, Nov. 2018. doi: [10.1007/s10207-017-0398-5](https://doi.org/10.1007/s10207-017-0398-5).
- [83] M. Kanwal and S. Thakur, "An APP based on static analysis for Android ransomware," in *Proc. Int. Conf. Comput., Commun. Automat. (ICCCA)*, May 2017, pp. 813–818.
- [84] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, and E. Kirda, "Cutting the gordian knot: A look under the hood of ransomware attacks," in *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2015, pp. 3–24. doi: [10.1007/978-3-319-20550-2\\_1](https://doi.org/10.1007/978-3-319-20550-2_1).
- [85] J. A. Gómez-Hernández, L. Álvarez-González, and P. García-Teodoro, "R-locker: Thwarting ransomware action through a honeypot-based approach," *Comput. Secur.*, vol. 73, pp. 389–398, Mar. 2018.
- [86] *CryptoStopper: Software to Detect and Stop Ransomware*. Accessed: Aug. 5, 2019. [Online]. Available: <https://www.watchpointdata.com/cryptostopper/>
- [87] (2017). *Ransomware Bundle v1.2.6*. Accessed: Apr. 2018. [Online]. Available: <https://docs.extrahop.com/current/walkthrough-bundle-ransomware/#appendix>
- [88] K. Cabaj and W. Mazurczyk, "Using software-defined networking for ransomware mitigation: The case of cryptowall," *IEEE Netw.*, vol. 30, no. 6, pp. 14–20, Nov./Dec. 2016.
- [89] M. M. Ahmadian, H. R. Shahriari, and S. M. Ghaffarian, "Connection-monitor & connection-breaker: A novel approach for prevention and detection of high survivable ransomsware," in *Proc. 12th Int. Iranian Soc. Cryptol. Conf. Inf. Secur. Cryptol. (ISCISC)*, Sep. 2015, pp. 79–84. doi: [10.1109/iscisc.2015.7387902](https://doi.org/10.1109/iscisc.2015.7387902).
- [90] D. Novakovic, "Umbrella, the Cisco's software to increase protection against ransomsware," Cisco Systems, San Jose, CA, USA, Tech. Rep., 2016. [Online]. Available: [https://www.cisco.com/c/dam/m/hr/training-events/2017/cisco-connect/pdf/Introducing\\_Cisco\\_Umbrella\\_for\\_cloud\\_based\\_threat\\_protection.pdf](https://www.cisco.com/c/dam/m/hr/training-events/2017/cisco-connect/pdf/Introducing_Cisco_Umbrella_for_cloud_based_threat_protection.pdf)
- [91] (May 2016). *TeslaCrypt Ransomware*. Accessed: Aug. 5, 2019. [Online]. Available: <https://www.knowbe4.com/teslacrypt-ransomware>
- [92] (Jul. 2016). *Ransomware Authors Flunk Again and Again*. Accessed: Jul. 4, 2019. [Online]. Available: <https://resources.infosecinstitute.com/ransomware-authors-flunk-again-and-again/#gref>
- [93] (2016). *No More Ransom Project*. Accessed: Apr. 2018. [Online]. Available: <https://www.nomoreransom.org>
- [94] E. Kolodenker, W. Koch, G. Stringhini, and M. Egele, "PayBreak: Defense against cryptographic ransomware," in *Proc. ACM Asia Conf. Comput. Commun. Secur. ASIA CCS*, 2017, pp. 499–611. doi: [10.1145/3052973.3053035](https://doi.org/10.1145/3052973.3053035).
- [95] H. Kim, D. Yoo, J. S. Kang, and Y. Yeom, "Dynamic ransomware protection using deterministic random bit generator," in *Proc. IEEE Conf. Appl., Inf. Netw. Secur. (AINS)*, Nov. 2017, pp. 64–68.
- [96] J. Huang, J. Xu, X. Xing, P. Liu, and M. K. Qureshi, "FlashGuard: Leveraging intrinsic flash properties to defend against encryption ransomware," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 2231–2244.
- [97] F. Quinkert, T. Holz, K. S. M. T. Hossain, E. Ferrara, and K. Lerman, "RAPTOR: Ransomware attack PredicTOR," Mar. 2018, *arXiv:1803.01598*. [Online]. Available: <https://arxiv.org/abs/1803.01598>
- [98] F. Mercaldo, V. Nardone, A. Santone, and C. A. Visaggio, "Ransomware steals your phone. Formal methods rescue it," in *Proc. Int. Conf. Formal Techn. Distrib. Objects, Compon., Syst.* Heraklion, Greece: Springer, 2016, pp. 212–221.
- [99] M. Baykara and B. Sekin, "A novel approach to ransomware: Designing a safe zone system," in *Proc. 6th Int. Symp. Digit. Forensic Secur. (ISDFS)*, 2018, pp. 1–5.
- [100] Y. Jin, M. Tomoishi, S. Matsuura, and Y. Kitaguchi, "A secure container-based backup mechanism to survive destructive ransomware attacks," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, 2018, pp. 1–6.
- [101] 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. (NCA)*, Oct./Nov. 2017, pp. 1–8.
- [102] (2017). *QuorumOnQ Ransomware Edition*. Accessed: May 2018. [Online]. Available: <https://quorum.com/resources/onq-ransomware-edition>
- [103] 2017. *Sophos Intercept X*. Accessed: Jun. 2017. [Online]. Available: <https://www.sophos.com/en/products/intercept-x.aspx>



**EDUARDO BERRUETA** graduated in telecommunication engineering from the Public University of Navarre (UPNA), Spain, in 2016, and the M.Sc. degree, in 2018, where he is currently pursuing the Ph.D. degree with the Telecommunications, Networks and Services Research Group. Previously, he attended the University of Turin for completing his thesis on software-defined networking. In 2016, he held a scholarship on the Automatics and Computing Department. In 2017 and 2018, he was a Research Assistant with the Telecommunications, Networks and Services Research Group, UPNA, while he completed the M.Sc. degree in telecommunication engineering.



**DANIEL MORATO** received the M.Sc. degree in telecommunication engineering and the Ph.D. degree from the Public University of Navarre, Spain. In 2002, he was a Visiting Postdoctoral Fellow at the Electrical Engineering and Computer Sciences Department, University of California, Berkeley, CA, USA. Since 2006, he has been with the Department of Automatics and Computing, Public University of Navarre, as an Associate Professor. In 2014, he became a member of the Institute of Smart Cities. His research interests include high-speed networks, performance and traffic analysis of the Internet services, and network monitoring.



**EDUARDO MAGAÑA** received the M.Sc. and Ph.D. degrees in telecommunications engineering from the Public University of Navarra, Pamplona, Spain, in 1998 and 2001, respectively. Since 2005, he has been an Associate Professor with the Public University of Navarra. In 2002, he was a Postdoctoral Visiting Research Fellow at the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA, USA. His main research interests are network monitoring, traffic analysis, and performance evaluation of communication networks.



**MIKEL IZAL** received the M.Sc. and Ph.D. degrees in telecommunication engineering, in 1997 and 2002, respectively. In 2003, he worked as a Scientific Visitant with Institute Eurecom, Sophia-Antipolis, France, performing measures in network tomography and peer-to-peer systems. Since then, he has been with the Department of Automatics and Computing, Public University of Navarre, where he is currently an Associate Professor. His research interests include traffic analysis, network tomography, high speed next generation networks, and peer to peer systems.

• • •