Machine Learning for Cyber Security

De Gruyter Series on the Applications of Mathematics in Engineering and Information Sciences

Edited by
Mangey Ram

Volume 15

Machine Learning for Cyber Security

Edited by
Preeti Malik, Lata Nautiyal and Mangey Ram

DE GRUYTER

Editors
Dr. Preeti Malik
Graphic Era University
CSIT Block
Bell Road, Clement Town
Dehradun 248001
Uttarakhand
India
preetimalik@geu.ac.in
preetishivach2009@gmail.com

Dr. Lata Nautiyal
University of Bristol
Merchant Venturers Building
Woodland Road
Clifton
Bristol BS8 1UB
Great Britain
lata.nautiyal@bristol.ac.uk

Prof. Dr. Mangey Ram
Department of Mathematics
Computer Sciences and
Engineering
Graphic Era University
566/6 Bell Road
Clement Town, Dehradun 248002
Uttarakhand
India
drmrswami@yahoo.com

ISBN 978-3-11-076673-8
e-ISBN (PDF) 978-3-11-076674-5
e-ISBN (EPUB) 978-3-11-076676-9
ISSN 2626-5427

Library of Congress Control Number: 2022942864

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the internet at http://dnb.dnb.de.

Cover image: MF3d/E+/Getty Images
Typesetting: Integra Software Services Pvt. Ltd.
Printing and binding: CPI books GmbH, Leck

www.degruyter.com

Preface

Cyber threats today are one of the expensive losses that an organization can face. Today, it is impossible to deploy effective cybersecurity technology without relying heavily on advanced techniques like machine learning and deep learning. Cybersecurity is a growing challenge in the era of Internet. This book addresses questions of how machine learning methods can be used to advance cybersecurity objectives, including detection, modeling, monitoring, and analysis of as well as defense against various threats to sensitive data and security systems. Filling an important gap between machine learning and cybersecurity communities, it discusses topics covering a wide range of modern and practical machine learning techniques, frameworks, and development tools to enable readers to engage with the cutting-edge research across various aspects of cybersecurity. The book focuses on mature and proven techniques, and provides ample examples to help readers grasp the key points. This cybersecurity book presents and demonstrates popular and successful artificial intelligence approaches and models that you can adapt to detect potential attacks and protect your corporate systems.

This book will assist readers in putting intelligent answers to current cybersecurity concerns into practice and in creating cutting-edge implementations that meet the demands of ever-more complex organizational structures. By the time you finish reading this book, you will be able to create and employ machine learning algorithms to mitigate cybersecurity risks.

https://doi.org/10.1515/9783110766745-202

Contents

List of contributors

1. Preeti Malik
Graphic Era Deemed to be University,
Dehradun, India
Email: preetishivach2009@gmail.com

2. Varsha Mittal
Graphic Era Deemed to be University,
Dehradun, India
Email: var.aadi@gmail.com

3. Mohit Mittal
INRIA Labs, France
Email: mittal.mohit02@gmail.com

4. Kamika Chaudhary
MB Govt. PG College, Haldwani, India
Email: kamikaduhoon@gmail.com

5. Abdul Rahman
Middlesex University Dubai, Dubai, UAE

6. Krishnadas Nanath
Middlesex University Dubai, Dubai, UAE
Email: username.krishna@gmail.com

7. Kiran Aswal
Gurukul Kangri Viswavidyalaya,
Dehradun Campus, Uttarakhand, India
Email: kiranaswal1984@gmail.com

8. Dinesh C. Dobhal
Graphic Era Deemed to be University,
Dehradun, Uttarakhand, India
Email: dineshdobhal@gmail.com

9. Umesh K. Tiwari
Graphic Era Deemed to be University,
Dehradun, Uttarakhand, India
Email: umeshtiwari22@gmail.com

10. Samuel Wedaj Kibret
Indian Institute of Technology Delhi,
New Delhi, India
Email: samuel.wed@cse.iitd.ac.in,
samuel_wed@yahoo.com

11. Sangeeta Mittal
Jaypee Institute of Information Technology,
Noida, Uttar Pradesh, India
Email: sangeeta.mittal@jiit.ac.in

12. Vasu Thakur
Department of Computer Science and
Engineering,
Roorkee Institute of Technology,
Roorkee, India

13. Vikas Kumar Roy
Department of Computer Science and
Engineering,
Roorkee Institute of Technology,
Roorkee, India
Email: royvikas1610@gmail.com

14. Nikhil Baliyan
Department of Computer Science and
Engineering,
Roorkee Institute of Technology, Roorkee,
Uttarakhand, India

15. Nupur Goyal
Department of Mathematics,
Graphic Era Deemed to be University,
Dehradun, Uttarakhand,
India
E-mail: nupurgoyalgeu@gmail.com

https://doi.org/10.1515/9783110766745-204

16. Rahul Nijhawan
Department of Computer Science and Engineering,
University of Petroleum and Energy Studies,
Dehradun, Uttarakhand, India
E-mail: rahul.nijhawan@ddn.upes.ac.in

17. Ashwini Kumar Singh
Department of Informatics, King's College London, London, United Kingdom

18. Rohit Nautiyal
University of Surrey, Guildford, United Kingdom

19. Swati Rawat
Quantum University, Roorkee, Uttarakhand, India
Email: swateerawat06@gmail.com

Editor's biography

Dr. Preeti Malik is associated as assistant professor with Graphic Era University, Dehradun, India, since 2016. She has acquired her doctorate in 2017 from Gurukul Kangri University, Haridwar, India, in mobile agent fault tolerance and security. She has published various research papers in reputed journals. Her research interests include cybersecurity, requirement engineering, and software reliability. She has published more than 15 papers in reputed journals. She has authored a textbook *Algorithms*, which was published by De Gruyter publishers, Germany.

Dr. Lata Nautiyal is associated as research associate with the University of Bristol, Bristol, UK. She is currently working on cybersecurity body of knowledge (CYBOK). She has acquired her doctorate in 2016 from Gurukul Kangri University, Haridwar, India, in component-based software engineering. She worked as an assistant professor with Graphic Era University Dehradun, India, for 13 years. During this period she was associated with academic and also research. She has published various research papers in reputed journals. Her research interests include cybersecurity, software testing, requirement engineering, and reliability.

Prof. Dr. Mangey Ram received his Ph.D. major in mathematics and minor in computer science from G. B. Pant University of Agriculture and Technology, Pantnagar, India. He has been a faculty member for around 12 years and has taught several core courses in pure and applied mathematics at undergraduate, postgraduate, and doctorate levels. He is currently a *research professor* at Graphic Era (Deemed to be University), Dehradun, India. Before joining the Graphic Era, he was a deputy manager (probationary officer) with Syndicate Bank for a short period. He is editor in chief of *International Journal of Mathematical, Engineering and Management Sciences*, book series editor with Elsevier, CRC Press-A Taylor and Frances Group, De Gruyter Publisher Germany, River Publisher, USA, and the guest editor and member of the editorial board of various journals. He has published 225 plus research publications in IEEE, Taylor & Francis, Springer, Elsevier, Emerald, World Scientific, and many other national and international journals and conferences. His fields of research are reliability theory and applied mathematics. Dr. Ram is a senior member of the IEEE, life member of Operational Research Society of India, Society for Reliability Engineering, Quality and Operations Management in India, and Indian Society of Industrial and Applied Mathematics. He has been a member of the organizing committee of a number of international and national conferences, seminars, and workshops. He has been conferred with *Young Scientist Award* by the Uttarakhand State Council for Science and Technology, Dehradun, in 2009. He has been awarded the *Best Faculty Award* in 2011, *Research Excellence Award* in 2015, and recently *Outstanding Researcher Award* in 2018 for his significant contributions in academics and research at Graphic Era Deemed to be University, Dehradun, India.

https://doi.org/10.1515/9783110766745-205

Preeti Malik*, Varsha Mittal, Mohit Mittal, Kamika

Differential privacy: a solution to privacy issue in social networks

Abstract: The privacy of social network data is becoming increasingly important, threatening to limit access to this lucrative data source. The topological structure of social networks can provide useful information for income production and social science research, but it is challenging to ensure that this analysis does not breach individual privacy. Differential privacy is a prominent privacy paradigm in data mining over tabular data that employs noise to disguise individuals' contributions to aggregate findings and provides a very exceptional analytical guarantee that individuals' existence in the data-set is hidden. Because social network analysis has multiple applications, it opens up a new field for differential privacy applications. This article provides a thorough examination of the fundamental principles of differential privacy and their applications in computing.

Keywords: Differential Privacy, global sensitivity, smooth sensitivity, degree distribution

1 Social media and its popularity

The growth of social media began in 1996 with the debut of the networking site Bolt (now closed).[1] Soon later, in 1997, Six Degrees was launched, allowing users to add friends and establish profiles. Following that, programs such as AOL Instant Messenger, Live Journal, and Friendster were created, all of which helped pave the way for Facebook to launch in 2004. Every day, more people are using social media. The number of active social media users worldwide reached 4.48 billion in 2021, an increase of 13.13% from 3.69 billion in 2020. In 2015, there were just 2.07 billion users, suggesting a 115.59% increase in just 6 years.

1 https://backlinko.com/social-media-users. On: 25-2-2022.

***Corresponding author: Preeti Malik**, Graphic Era Deemed to be University, Dehradun, India, e-mail: preetishivach2009@gmail.com
Varsha Mittal, Graphic Era Deemed to be University, Dehradun, India
Mohit Mittal, INRIA Labs, France
Kamika, MB Govt. PG College, Haldwani, India

https://doi.org/10.1515/9783110766745-001

1.1 Pandemic marketing update

In July 2020, DataReportal produced a unique report that examines changes in social media activity at the commencement of the COVID-19 lockdown period, in addition to usual enquiries. The amount of Internet and digital activities has increased dramatically (see Figure 1).

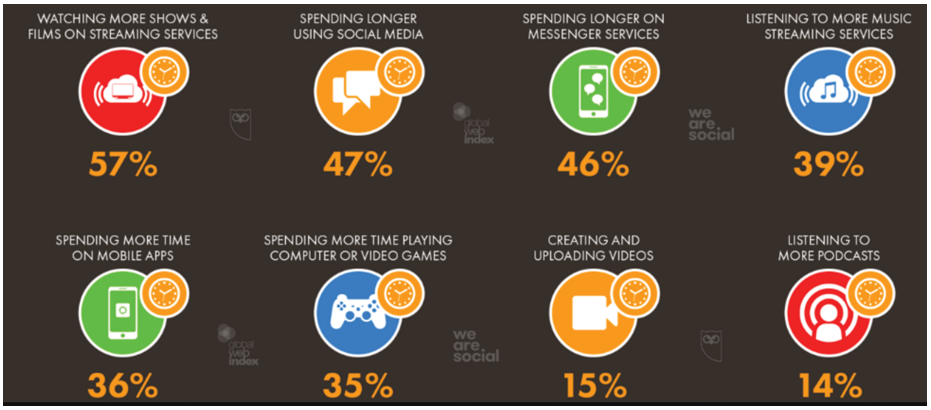


Figure 1: Impact of COVID-19 on online activities.[2]

1.2 Pros and cons of using social media

1.2.1 Pros

There are several more advantages of using social media:

- Digital media knowledge: It allows your kid to explore and experiment on social media. It also aids them to get the material and proficiencies they need to adore online events while escaping from online risks.
- Cooperative learning: Your kid may exchange educational information on social media.
- Creativity: Your youngsters may express themselves through their profile pages, images, and videos.

2 https://raisingchildren.net.au/teens/entertainment-technology/digital-life/social-media#: ~:text = Social%20media%3A%20risks,-Social%20media%20can&text = uploading%20inappropriate%20content%2C%20like%20embarrassing,much%20targeted%20advertising%20and%20marketing] on 25-2-2022.

– Mental health and well-being: Interacting with people and friends on social media provides an emotion of belonging and connection in your kid.

1.2.2 Cons

Social media may sometimes be dangerous. The dangers for your kids include:
– Uncovering aggressive or distressing information, like harsh, offensive, violent, or sexual remarks or snaps.
– Sharing wrong content, for instance, snaps or videos that are uncomfortable or suggestive.
– Sharing personal information on social media with strangers, for example, contact number, birth date, or addresses. Privacy settings can limit who can view information about your kids, such as their name, age, and where they reside. One can misuse this information.
– One can become victim of cyberbullying.

2 Social network analysis

Disease transmission, emotional contagion, and professional mobility are all examples of critical societal concerns that may be discovered via social network research [1, 5]. Social networks are designed to distribute data without revealing personal information due to the requirement for scientific study and data exchange. The original data can be disturbed or encrypted, or anonymous processing can be performed before releasing the data [2–4].

The phrase “privacy” is loaded, since it means different meaning for different people. Edge weights in social networks may indicate the frequency of contact, the cost of economic exchange, the closeness of a connection, and other factors that are linked to sensitive data. An intelligence network is a good example, where edge weights represent the frequency of communication between two organizations. Excessive communication might indicate an issue. A commercial trade network is another illustration, where edge weights represent the price of a transaction between two businesses. Due to the severe rivalry, most managers would be hesitant to give a business secret to their competitors. Our objective is to prevent edge weight leaking in social networks while retaining as much usefulness as feasible.

Dalenius [6] first diagnosed privacy protection issue in the late 1970s. According to Dalenius, privacy protection aimed at preventing any user either legitimate or spurious, from accessing original data of any individual while accessing the database. A number of solutions have been proposed by researchers which are based on this idea, including k-anonymity [7], l-diversity [8], t-closeness [9], and (α, k)-

anonymity [10]. Though all these models protect against a certain form of assault and are unable to fight against newly invented attacks, the security of the model is based on the hypothesis of some specific background information of an attacker, which is a primary source of this flaw. Nonetheless, enumerating all conceivable sorts of background information of attacker may have been very hard. As a result, a model that preserves privacy while ignoring background knowledge is very desired.

3 Privacy breaches in social networks

Defining the term *privacy breach* is crucial [11]. When a bit of delicate information of an individual is given to an enemy or to someone having the objective of damaging privacy, it is called a privacy breach. Identity disclosure and attribute disclosure are the two forms of privacy breaches that have traditionally been researched. In the framework of social networks, we explore these two forms. We also discuss two other forms of network data disclosures: social link and affiliation link disclosure.

3.1 Identity disclosure

When a challenger is capable of discovering the mapping from a social network profile $\boldsymbol{p}$ to a particular real-world entity $\boldsymbol{i}$, identity exposure happens. Let us analyze three issues about $\boldsymbol{i}$'s identity in which an opponent would be interested before we can establish a formal definition of identity disclosure. These definitions (see Table 1) have been taken from Zheleva and Getoor [11].

Table 1: Query definitions.

Definition 1 (Mapping query). In a set of individual profiles (P) in a social network G, find which profile p maps to a particular individual i. Return p.
Definition 2 (Existence query). For a particular individual $\boldsymbol{i}$, find if this individual has a profile $\boldsymbol{p}$ in the network G. Return true or false.
Definition 3 (Co-reference resolution query). For two individual profiles $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$, find if they refer to the same individual $\boldsymbol{p}$. Return true or false.

To put it another way, identity disclosure means that the attacker can properly and confidentially answer the mapping question. This is difficult to do if the attacker knows unique properties of individual $\boldsymbol{p}$ that may be matched with observable attributes of profiles in $\boldsymbol{P}$. One technique to formalize identity disclosure for an individual $\boldsymbol{p}$ is to assign a random variable $\boldsymbol{vp}$ that spans all of the network profiles. We suppose that the attacker knows how to compute:

$$\Pr(vp = p_i)$$

where the probability of each profile $\boldsymbol{p_i}$ belongs to person $\boldsymbol{p}$. Furthermore, we insert a dummy profile $\boldsymbol{p}_{\text{dummy}}$ into the network to absorb the chance that person $\boldsymbol{p}$ does not have a profile in the network. We suppose that $\boldsymbol{p}$ has only one profile, and that in $\boldsymbol{P}$ $\boldsymbol{p}_{\text{dummy}}$, p's actual profile is v. To represent the likelihood that v_i corresponds to p, we use the shorthand $\Pr p(v_i) = \Pr(vp = v_i)$; $\Pr p$ gives a mapping $\Pr p$: V v_{dummy} R. We leave it up to the opponent to figure out how $\Pr p$ is built. There are many researchers who work in social network privacy and focused on identity disclosure [12–22].

3.2 Attribute disclosure

There are three sorts of personal attributes, according to a prevalent theory in the privacy literature:

- *Identifying attributes* – qualities that uniquely identify a person, such as a social security number (SSN).
- *Quasi-identifying attributes* – a set of traits that may be used to uniquely identify a person, for example, person's name and his/her address.
- *Sensitive attributes* – characteristics that an individual would like to keep private, like political affiliation.

When an attacker is capable of identifying the value of a confidential attribute of a user that the user wanted to keep secret, this is known as attribute disclosure. This characteristic can be associated with the node itself, its connections, or its affiliations. We will talk about the node's characteristics here without losing generality. We can also redefine attribute disclosure like, suppose each sensitive characteristic v.as for profile v is connected with a random variable v.as that spans the whole range of v.as values. Allow v.as to have its correct value of v.a. For each potential value v.a, we assume that the attacker can interpret the set of possible confidential attribute values to probabilities, Pra(v.as = v.a): v.a R. It is worth noting that the mapping for each node/profile might be different.

If an opponent has access to identifying attributes in a social network, answering the identity mapping question becomes straightforward, and identity revelation with confidence 1 is possible. If a profile has an SSN, for example, recognizing the actual user behind the profile is simple because persons and their SSNs have a one-to-one correspondence. As a result, identifiable characteristics must be deleted from profiles to prevent identity revelation.

A set of qualities accepted as quasi-identifying attributes can sometimes give clue of the identity disclosure. What defines quasi-identifying characteristics varies depending on the situation. For instance, according to the 1990 US Census, 87% of people may be individually recognized based on their birth date, gender, and postal

code [23]. A combination of name and address of a person is another example of a quasi-identifier.

Matching records from disparate databases with quasi-identifying features can also lead to further privacy violations. For example, matching health insurance data of a person with that of public voter registration records can expose delicate information about voters' health if the identifying information is deleted. Sweeney was able to locate the governor of Massachusetts' medical records using this method [23].

Till date, a very few studies focused on attribute disclosure from the perspective of social and affiliation networks. The majority of researches [24–26] focus on how qualities can be predicted, with only a few focusing on how they can be safeguarded [27].

3.3 Social link disclosure

When an enemy discovers the presence of a delicate association between two users that they would prefer to keep secret from the community, this is known as social link disclosure. We suppose that an arbitrary variable $e_{i,j}$ is connected with the presence of a connection between two nodes n_i and n_j, and that an attacker has a method for allocating a probability to $e_{i,j}$, $\Pr(e_{i,j} = \text{true})$: $e_{i,j} \rightarrow R$, similar to the earlier types of leaks.

Social networks, communication data, medical data, and other data sources all contain examples of sensitive interactions. Based on a person's friendship links and the public likings of their friends, it may be feasible to deduce the person's own

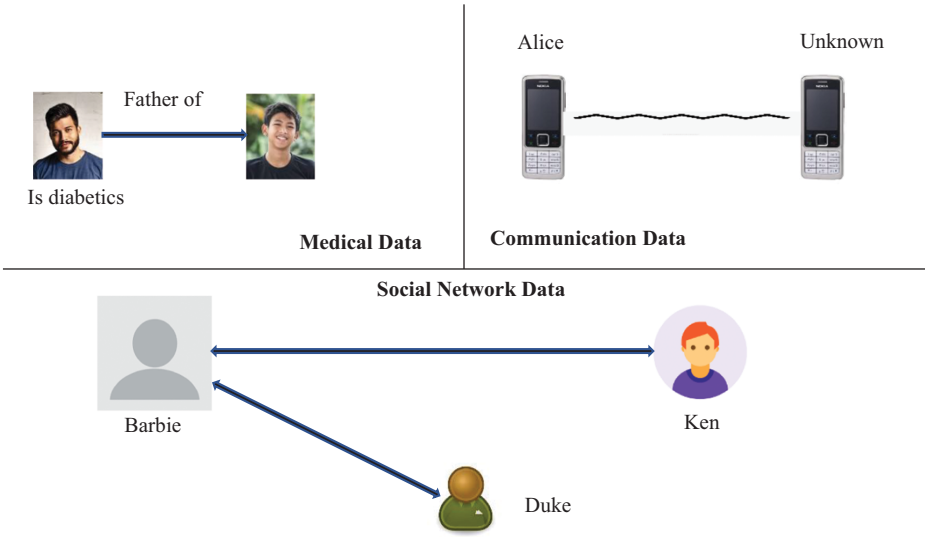


Figure 2: Sensitive link examples.

preferences from social network data. In mobile phone communication data, discovering that an anonymous person has made phone calls to a cell phone number of a recognized organization might compromise the unknown person's identity. Knowing the familial links between persons who have been detected with genetic illnesses and those who have not can assist to extrapolate the likelihood of healthy persons developing these disorders in hereditary disease data.

Researchers have looked into social network attacks that disclose sensitive linkages [28–31]. Figure 2 shows some examples of sensitive links. Recent research has also focused on sensitive edge features such as link strength [32, 33].

3.4 Affiliation link closure

One more type of relational data privacy infringement is affiliation link disclosure, which shows whether an individual fits in a certain affiliation group. It might also be delicate to determine if two users are members of the same group. This type of disclosure can lead to other three types of disclosures. As a result, keeping one's privacy requires concealing one's affiliations.

Again, let us suppose that there is an arbitrary variable $\boldsymbol{e_{p,h}}$ coupled with the presence of an affiliation link between a profile $\boldsymbol{p}$ and a group $\boldsymbol{h}$, and that an attacker has a method to compute the probability of $e_{p,h}$, $\Pr(e_{v,h} = \text{true})$: $e_{v,h} \rightarrow R$.

It is possible that one form of expose will give hint of another. Wondracek et al. [19], for example, demonstrate a de-identification attack in which the revelation of an affiliation connection can lead to the identity of a seemingly unidentified Internet client. An attacker begins the assault by scanning a social networking Internet site and gathering facts and data about its users' affiliations in online social groups. The identity of social network users is considered to be known. As per the information gathered, each user who belongs to minimum one group has a group signature, which is a list of the groups to which he belongs. The adversary then performs a history theft attack (for additional information on the assault, see [19]), which captures the target Internet user's online surfing history.

Search data is an illustration of affiliation connection disclosure which lead to identity exposure. If we believe that users who submit search questions to a search engine are members of the social network, and that the search questions they submit represent affiliation groups, at that point revealing the relationships between query submitted and the user can aid the attacker in identifying members of the network. Users engage with search engines in an unrestricted manner, disclosing a great deal of personal data in the content of their requests. In 2006, an Internet service provider, AOL, provided an "anonymized" sample of nearly half a million customers and their queries to the AOL search engine, causing a scandal. The release was well-meant, with the goal of augmenting search ranking studies with real-world data.

One of the issues with the provided data was that, despite being in table format, the items were not self-contained. Shortly after the data was released, reporters from the *New York Times* connected 454 search requests made by the same person, which revealed enough personal information to identify that person – Thelma Arnold, a 62-year-old widow from Lilburn, Georgia [34]. Her inquiries included information on others with the same last name as hers, retirement, and her location.

As shown in a guilt-by-association assault [35], affiliation link revelation can also lead to attribute disclosure. This attack implies that there exist groups of users with the same sensitive attribute values; therefore, retrieving one user's sensitive value and the affiliation of another user to the group can assist in recovering the sensitive value of the second user. This exploit was used to learn about users' downloading patterns on the BitTorrent file-sharing network [36]. Communities were discovered through social connections, and watching only one person in each group was enough to deduce the interests of the others. The sensitive attribute that consumers would wish to keep hidden in this scenario is whether or not they are violating copyrights. This technique has also been used in a phone network to identify fake callers [35]. Data anonymization was used by Cormode et al. [37] to prevent affiliation connection revelation.

4 Privacy preservation methods

4.1 Anonymization

The practice of deleting personally identifying information from a dataset in order to safeguard people's privacy is known as data anonymization. It allows data users and owners to securely share data for data analysis, decision-making, research, and other purposes while maintaining the anonymity of individuals whose information is contained in the dataset. The data is modified by the curator (the person who gathered it) by eliminating particular identifiers such as name, security number, address, and phone number. Even if the specific identifiers are removed, the availability of an individual's background data makes it easier for the attacker to reidentify individuals by linking the released data, making data publication without revealing personal information extremely difficult [38]. It is difficult for the owners to regulate how the data is modified once it has been provided to a third party. By reidentifying governor William Weld's medical information, Latanya Sweeney, an MIT graduate student in computer science, demonstrated that individual's information in anonymously published data may be reidentified by matching the disclosed data to publicly available data [38].

4.2 *k*–Anonymity

To address the disadvantages of basic data anonymization, academics have developed a number of privacy-preserving approaches. The *k*-anonymity approach is one of the most common ways to protect one's privacy. Using quasi-identifiers to prevent record linking, the concept of *k*-anonymity was suggested by Samarati and Sweeney [39]. Its goal is to disclose data with a scientific assurance that a specific individual's data cannot be uniquely identified while the data may be used in a reasonable manner. The feature of a *k*-anonymized data collection is that each individual in the record is comparable to at least another $k - 1$ other records on the potentially identifiable factors. The level of data protection on inference by linking is characterized as *k*-anonymity. It prohibits the published data from being linked to other sources of information (background information). *k*-Anonymity, on the other hand, does not ensure privacy. Two assaults were employed by Machanavajjhala et al. [40] to demonstrate how *k*-anonymity does not ensure privacy.

4.3 Homogeneity attack

When there is minimal variance in the sensitive characteristics, the adversary can detect the value of the sensitive attribute for that collection of *k*-records using the homogeneity attack. For example, a politician seeking election to a position in state government uses his/her opponent's medical background to show the public that his/her opponent is unable to fulfill his/her responsibilities as an agent of the state owing to his/her medical issues. He/she will have to use the hospital's disclosed data from the three-anonymized table to look for his/her opponent's medical information. Despite the fact that the data is most likely a three-anonymized table, he/she can detect what disease his/her opponent has because there are few contrasts (low variation) in the sensitive data because he/she has some knowledge about him/her. For example, if he/she knows that the patient is a 25-year-old American who resides in postal division 11003, based on this information, he/she deduces that his/her competitor has heart disease.

4.4 Background knowledge attack

Background knowledge is used by the adversary in this attack, and we will prove that *k*-anonymity does not ensure privacy against background knowledge assaults. A woman whose colleague's father is ill, for example, must understand the nature of the illness. She is aware that her coworker's father is elderly and Mexican, so she may deduce that he is suffering from either vitamin D deficiency or Alzheimer's disease. Nonetheless, it is recognized that, for the most part, Mexicans are unaffected

by vitamin D deficiency. Alzheimer's disease is a prevalent neurological disease that affects the elderly. As a result, she quickly deduces that her colleague's father suffers from Alzheimer's disease. She uses her previous knowledge to figure out what ailment her colleague's father is suffering from. As can be seen from the earlier instances, *k*-anonymity does not ensure privacy preservation.

4.5 *l*-Diversity

Although *k*-anonymity protects privacy from record identification, it is not always successful in preserving privacy from inference assaults on sensitive characteristics. Machanavajjhala et al. [40] established a new concept known as *l*-diversity, which states that any tuple with identical quasi-identifiers must have at least *l*-varied well-represented values for the sensitive attribute. According to Machanavajjhala et al., an equivalence class has *l*-diversity if at least *l* well-represented values for the sensitive characteristic are present. An *l*-diverse table is one that has equivalence classes that are all *l*-diverse. *l*-Diversity, in summary, is a paradigm that encourages intragroup variability of sensitive qualities by at least "*L*" distinct values. Though *l*-diversity was proposed to overcome *k*-problems, anonymity such as attribute linking, Li et al. [41] have shown that it does not adequately address the issue of attribute disclosure. They used two assaults to illustrate this: similarity attack and skewness attack, when the equivalence class contains distinct but semantically similar sensitive attribute values. Because the real population's distribution differs from the dataset, *l*-diversity fails to avoid attribute disclosure in the first situation. As a result, the sensitive attribute distribution within the equivalence class differs from the real population, resulting in attribute disclosure. The adversary estimates the value of a sensitive attribute in the latter case by first tying it to another sensitive attribute.

4.6 *t*-Closeness

To prevent the limitations of *l*-diversity, Li et al. [41] proposed a notion of privacy called *t*-closeness. The formal definition of *t*-closeness given by Li et al. is stated further.

> An equivalence class is said to have t-closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute in the whole table is no more than a threshold t. A table is said to have t-closeness if all equivalence classes have t-closeness.

An equivalence class is a set of data that have the same values for their quasi-identifiers.

5 Differential privacy

Cynthia Dwork of Microsoft Research Labs invented differential privacy [42]. It is a mathematical promise of privacy that sufficiently well-privatized queries may meet, rather than a specific approach or procedure. Consider the following scenario in social science research: Individual data from the surveys are combined into a dataset and some analysis is done over it; the analysis may be privatized by injecting random noise; and the final privatized result is published to the wider public. Differentially private inquiries provide survey participants with a mathematical guarantee that the results will not expose their involvement in the survey.

The aim behind differential privacy is to incorporate a controlled amount of statistical noise into results of the query to disguise the impact of a single individual being added or removed from a dataset. It means, when an attacker queries two nearly identical datasets (with only one record difference, for instance), the outcomes are differentially privatized so that an attacker would not be able to discover any new information about an individual with a high probability.

Let f stand for a query function that will be evaluated on the dataset D. We want an algorithm A to run on a dataset D and output A(D), with A(D) being f(D) with a regulated amount of random noise added. The purpose of differential privacy is to get A(D) as near to f(D) as feasible to maintain data usefulness while also protecting the privacy of the dataset's entities.

Differential privacy is primarily concerned with adversarial attacks that query databases that differ only by a few elements. Differential privacy is divided into two types: unbounded and limited, as defined by the concept of nearby datasets [43]. Unbounded means that for two datasets D and D_0, D_0 may be produced by adding or subtracting a tuple from D. It is said to be bounded if D_0 may be produced by altering the value of a tuple from D, that is, bounded nearby datasets have the same size, but unbounded neighboring datasets are one size apart. Although the presentation of query results for unbounded and limited nearby datasets differs slightly, the concepts of constructing and assessing differential privacy methods remain the same. As a result, we use both types of nearby datasets in this chapter to demonstrate the introduced differential privacy techniques.

Definition 1 ([44]). A randomized algorithm A is ϵ-differentially private if for any two neighboring datasets D and D_0, and any subset S of possible outputs of A,

$$\Pr[A(D) \in S] \leq e^{\epsilon} \Pr[A(D_0) \in S]$$

where $\epsilon \geq 0$ is a parameter called privacy budget.

The privacy budget ϵ is often a tiny positive real number that represents the degree of privacy protection that algorithm A can give. For example, if $\epsilon = 0.01$, $e^{0.01} \approx 1.01$; and 0.01-differential privacy assures that A(D) and $A(D_0)$ distributions are highly close and nearly indistinguishable. The higher the amount of privacy

preserved, the lower the value of ϵ because more noise must be supplied, a smaller value gives more privacy preservation at the cost of reduced data accuracy. When ϵ = 0, the level of privacy protection is at its highest, that is, "complete" protection. The approach produces two outcomes with similar distributions in this situation; however, the accompanying findings provide no valuable information about the dataset. As a result, the value of ϵ should strike a balance between privacy and data utility. Usually requires extremely tiny values like 0.01, 0.1, or ln 2, ln 3 [42] in real applications. In some cases, computing ϵ-differential privacy can be difficult. An extended idea of differential privacy is proposed to aid approximation.

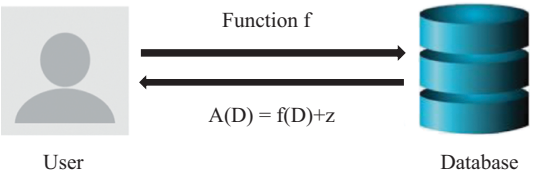


Figure 3: Adding noise to the result.

A(D) = f(D) + z, as shown in Figure 3, can be used to create differential privacy by introducing a suitable amount of noise to the result of the query. Large amount of noise can reduce data usefulness, and very less noise cannot provide enough privacy protection. The crucial parameter to estimate the degree of the additional noise is known as *sensitivity*, which indicates the biggest change in query results generated by either adding or removing any record in the dataset. As a result, the differential privacy model defines global sensitivity, local sensitivity, smoothing upper bound, and smoothing sensitivity.

5.1 Types of differential privacy

5.1.1 Node privacy

For a privatized query *PQ* node privacy is preserved when the differential privacy for each couple of graphs is satisfied and can be explained as follows:

Let G_1 and G_2 be two graphs with (V_1,E_1) and (V_2,E_2) as a set of vertices and edges, respectively, such that

$$|(V_1 \cup V_2)/(V_1 \cap V_2)| = 1$$

and

$$\{(E_1 \cup E_2)/(E_1 \cap E_2)\} = \{(u,v)|u = y \vee v = y\}$$

Here y is the node that exists in $(V_1 \cup V_2)/(V_1 \cap V_2)$ and the edge between nodes u and v is represented by (u,v). Node privacy assures fully protection to both participant and subjects. An attacker with R will be unable to determine whether or not a person y exists in the population. The queries we can compute are severely limited as a result of this.

In this type of differential privacy, neighboring graph G' of a given social network G is derived by removing or introducing a node and all edges incident to that node. The goal of node differential privacy is to preclude an attacker from identifying as if a specific node x exists in the graph. It ensures privacy for individuals and relationships at the same time, instead of a single relationship, at the expense of rigid query constraints and lower accuracy outcomes. Under node privacy, a differentially private algorithm must hide the worst-case disparity between neighboring graphs, which can be significant. For instance, consider a star graph where a node is connected to all nodes. In such scenario, the graph will have high sensitivity; moreover, adding a noise to such graph is more vivid. Because of its high sensitivity, it is utterly impossible for node privacy to provide correct network analysis, but required privacy protection can be obtained [45].

5.1.2 Edge privacy

To preserve edge privacy in a decentralized query, all couple of graphs that exist for the graphs G_1 and G_2 with (V_1, E_1) and (V_2, E_2) as a set of vertices and edges, respectively, should satisfy differential privacy and both G_1 and G_2 also fulfill the following property: $V_1 = V_2$ and $\{(E_1 \cup E_2)/(E_1 \cap E_2)\} = 1$.

To achieve edge privacy, a neighboring graph G' is obtained by removing or adding one edge from a social network graph G. It can be extrapolated to change up to k edges. Edge privacy prevents an attacker from learning about specific user relationships and also from identifying with high possibility whether two individuals are friends. It also denies the probability of the existence of a single node having k friendships with various nodes of the graph. In comparison to node privacy, this type of privacy can only protect information about user relationships [46]. Regardless of the fact that the associations between these nodes have been secured, nodes with higher degrees seem to have a greater impact on query results. However, this is sufficient for many applications and allows for the privatization of several types of queries than that of the severely restricted node privacy. For example, for preserving email relationships, the edge privacy is used by different researchers [47].

5.1.3 Out-link privacy

To preserve out-link privacy in a decentralized query, all couple of graphs that exist for the graphs G_1 and G_2 with (V_1, E_1) and (V_2, E_2) as a set of vertices and edges, respectively, should satisfy differential privacy, and both G_1 and G_2 also fulfill the following property: $V_1 = V_2$ and a node y exists in such a way that

$$\{(E_1 \cup E_2), /(E_1 \cap E_2)\} = \{(y \to v) | y \in V_1 \wedge v \in V_2 \text{ or } y \in V_2 \wedge v \in V_1\}$$

$y \to v$ denotes the directed link from y to v.

To achieve out-link privacy, a neighboring graph G' can be obtained from an SN graph G by either deleting all current out-links of a node y, or inserting a single or more than that out-link to a node with out-degree 0. Conceptually, the privacy standards used in out-link privacy are similar to the standards used in node privacy. As compared to node privacy, out-link privacy is feebler, but for some specific queries it shows better results than edge privacy [46].

This type of privacy can decrease the distinctive attributes of high-degree nodes, for example, a more popular person (high-degree node) can reject mutual friendships in query results even if many others still claim to be friends with this person. Out-link privacy enhances sensitivity computation and decreases the amount of injected noise required, and this would permit to perform certain queries that are otherwise impossible to perform with node and edge privacy. Degree distribution is a good example which shows that out-link privacy requires less noise.

5.1.4 Partition privacy

Various disjoint elements H_i comprise a partitioned graph G. To preserve partition privacy in a decentralized query, all couple of graphs that exist for the graphs G_1 and G_2 with (V_1, E_1) and (V_2, E_2) as a set of vertices and edges, respectively, should satisfy differential privacy, and both G_1 and G_2 also fulfill the following property: $G_1 = G_2 - H_i$ where $H_i \in G_2 \wedge H_i \notin G_1$ or $G_2 = G_1 - H_i$ where $H_i \in G_1 \wedge H_i \notin G_2$.

To achieve partition privacy, a neighboring graph G' can be obtained by inserting or removing a current subgraph from a social graph G. Rather than a connected social graph, most social structure queries are executed across a set of subgraphs [48]. A large social graph can be partitioned into multiple subgraphs using node characteristics like location, major, and occupational status, and each subgraph can be handled as a multiproperty data point. Then, removing or introducing a data point is similar to erasing or introducing a subgraph. As a result, the set of subgraphs, that is, different data points can be subjected to conventional differential privacy.

Partition privacy is more comprehensive than node privacy, in that it protects an entire social group rather than a single node.

6 Privacy attacks in social network

This section summarizes the different privacy attacks that frequently occur in social network. Privacy assaults cover a wide range of behaviors that expose sensitive information to individuals that should not have access to it. Inference attacks [49] are the attacks that compromise users' private information by analyzing contextual data, for instance, user professions or salary are the most significant type of privacy assault in online social networks (OSNs). In social networks, two types of inference assaults have been observed: private attribute inference [50–52] and user de-anonymization [18, 19, 53–58].

6.1 Private attribute inference

As the number of people using social networking sites continues to rise at a rapid rate, privacy and security concerns are becoming increasingly prevalent. Users' private characteristics can be deduced from their public activity on social media, even if they do not intend to reveal them. This type of privacy is called as private attribute inference. The goal of private attribute inference is to uncover a concealed value of the attribute that the user or service provider has purposefully hidden. In this type of attack, an attacker tries to spread the values for missing or incomplete data of publicly revealed attributes using the attribute information from social network. Any party (e.g., a malicious user, an OSN provider, an endorser, a data negotiator, or a monitoring agency) with an interest in users' confidential data could be the attacker. The attacker just has to obtain publicly available information from OSNs to carry out such privacy attacks. Aside from privacy issues, the implicit user characteristics can be cast off (by the invader or anyone who acquires the contingent user information from the attacker) to engage in a variety of security-sensitive activities, like phishing [59, 60] and attempting to compromise personal-information-based backup authentication [61]. Furthermore, an invader can utilize the inferred characteristic information to associate online users across numerous sites [62–65] or with offline information (such as records of voter registration which is publicly accessible) [66, 67], resulting in even greater security and privacy problems.

Friend-based and behavior-based attribute inference assaults are the two types of attacks now in use [50, 68–70]. Attacks on friends are predicated on the premise that you are, whom you know. They want to extrapolate features for a user gathered from multiple features extracted of the user's friends and the social structure among them. Homophily is the cornerstone of friendship-based assaults, which means that two linked users have comparable characteristics. For example, if more than 50% of a user's friends major in IT engineering at a particular university, the user is likely to major in IT engineering at that university as well. Behavior-based attacks refer qualities for a user grounded on the public traits of users who are

alike, and behavioral data is used to identify similarities between users [71–73]. These types of attacks are based on the concept that you are what you do. Users with the similar traits, in particular, have comparable interests, characteristics, and cultures, resulting in similar actions. For example, if a user loved music tracks, apps, and books on Google Play that were comparable to those loved by Indian users, the individual might belong to India.

6.2 User de-anonymization attack

The nodes in an anonymized graph and a reference graph are mapped with the legitimate user identities as inputs in user de-anonymization, allowing the users' characteristics to be redefined in the anonymized graphs [18, 19]. Different anonymization methods, like clustering, pseudonyms, graph amendment, and generalization, are used to conceal the personal distinguishable information after that a service provider usually releases an anonymized social network graph to various activists, like researchers, application developers, advertisers, and government agencies [53–57]. A reference graph can readily be created using information collected from various sources, for example, a distinct social network with overlapping participants with a publicly available social graph. In comparison to an anonymized social network graph, a reference graph typically has fewer node properties [58].

Backstrom et al. [74] reveal different types of active attacks on anonymized social networks' edge privacy. These active attacks presume that the attacker has the potential to modify the network before it is released. A malicious user selects a random set of users whose private information it intends to breach, and creates a small quantity of new user accounts including edges which are connected to the targeted users. Then, among the new accounts, a structure of links is obtained with the motive of achieving the anonymized graph structure. Both attacks rely on the creation of $O(\log N)$ new "sybil" nodes (the number of nodes is represented by N), whose outgoing edges aid in the quadratical reidentification of as many current nodes as possible.

The de-anonymization attacks are challenging to be conducted on large scale because of the given reasons. To begin with, they are limited to OSNs; constructing thousands of phony nodes in a phone call or real-world network is either too expensive or impractical. Even with OSNs, many operators (e.g., Facebook) examine the originality of email addresses and use other ways to verify the accuracy of supplied data since generating thousands of dummy nodes is a challenge.

Subsequently, the adversary has minimal rheostat over the edges that flow into the nodes he/she builds. A subgraph that does not have any incoming edges but have many outgoing edges will stick out as most authorized users will have no inducement to associate back with the sybil nodes. This could help the network operator figure out if the network has been hacked by a sybil attack. Other strategies for

detecting sybil assaults in social networks exist [75], such as spammer detection methods implemented by OSNs with unidirectional edges [76].

Another constraint of active attacks is that the usual social networks need a mutual relation before any information in any form is made available. Assuming that actual users do not link back to dummy users, the network does not show links from false nodes to real ones.

We believe that large-scale active attacks that necessitate the establishment of tens of thousands of sybil nodes are implausible. Active attacks can nevertheless be effective for discovering or manufacturing a small number of "seeds" that can be used to launch large-scale, passive privacy breaches.

Another type of de-anonymization attack is passive attacks in which a small group of users uses their knowledge of the network topology surrounding them to figure out where they are in the anonymized graph [76]. This assault is plausible, but it only works on a small scale: the cooperating users can only violate the privacy of some of their friends' users.

7 Application of differential privacy in social network analysis

The quantitative examination of data created by social network services using statistics, graph theory, and other methodologies is known as social network analysis. Some of the most common tasks in social network analysis includes degree distribution, edge weight analysis, triangle counting, k-star counting, and k-triangle counting. We examine a few frequently used strategies in social network analysis under differential privacy preservation in this section.

7.1 Degree distribution

A graph's degree distribution is a histogram that divides the nodes in the graph according to their degree; it is frequently used to characterize the primary structure of social networks with the motive of building graph models and comparing graphs. It reflects graph structure statistics and may have an impact on the entire graph operation process.

Despite the fact that degree distributions are depicted as histograms, node privacy has a high sensitivity since one node impacts numerous counts in the distribution. If a node is deleted from the graph, the degree of all connected nodes is reduced. A critical analysis shows that a node of degree d impacts not more than $2d + 1$ values in the histogram. In the adverse scenario, if a node with highest degree d is added or removed, it can modify only $2n + 1$ values, indicating that global

sensitivity is reliant on the number of nodes n in the graph. The degree histogram query is unsustainable for differential privacy protection under node privacy because n is unbounded.

It is possible to safeguard degree histogram queries applying differential privacy within edge privacy. Deleting one edge from the graph impacts at most four counts and modifies the degree of two nodes. Therefore, the sensitivity of k-edge privacy is $4k$. This k is a negligible amount of noise with a suitably large graph, resulting in data utility preservation.

For a degree histogram query, out-link privacy necessitates less noise. When only out-degrees are considered, eliminating one node's out-links from a graph changes one value in the histogram [48]. Under this privacy criterion, a node with a large value of degree may still leave traces of its appearance in the dataset through the friends' out-degree. Though there are a variety of possible descriptions for the graph having more than the expected degree among nodes, they could indicate new connections among the nodes, or may have friendships with persons who were not survey members. To use this susceptibility to anticipate the existence of high node with any accuracy, an attacker would need to have a near-complete understanding of the real SN [77].

7.2 Subgraph counting

The graph G is considered as an input graph and H is taken as a query graph, and the list of all isomorphic graphs of H in G is returned by subgraph counting query. Different examples of subgraphs are triangles, k-stars, k-triangles, and k-cliques. A k-star is made up of a center node that connects to k other nodes, a k-triangle is made up of k-triangles that share one common edge, and clique including k-vertices is called k-clique.

Subgraph counting queries involve varying levels of privacy and high global sensitivities. For attaining differential privacy, a considerable quantity of noise must be added, which may result in serious query result anomalies. As a result, the noise magnitude is usually determined by a smooth upper bound of the local sensitivity. Additionally, in the literature [78–80], truncation, ladder function, and Lipschitz extension were used to establish differential privacy while enhancing counting speed. In this section, we look at triangle, k-star, and k-triangle counting issues.

7.2.1 Triangle counting

To reflect the connectedness in SN, the concept of triangles is used as it occurs when two friends have mutual friends. Triangle counts are the most important

component in the clustering factor, which is a popular measure for characterizing and comparing graphs.

Under simple node privacy, it is not feasible to count the differentially private triangle. In the worst scenario, inserting a node to an n-dimensional graph (a graph with all conceivable edges) introduces $(n/2)$ new triangles. Because the alteration is proportional to the size of the network, the global sensitivity of query is unbounded: it is unfeasible to estimate a restricted global upper bound.

Because of the same reasons, for triangle counts, edge privacy is not possible. In the worst situation, consider a network having n nodes; if an edge is deleted, it can result in the removal of $n - 2$ triangles. Even though global sensitivity of the query of triangle counting is not bounded but for some specific graphs, its local sensitivity is bounded under edge privacy. Smooth sensitivity can be used to attain differential privacy [78, 81]. We briefly outline differentially private edge and node algorithms, and also additional ways for achieving differential privacy in triangle counting is reviewed.

Edge privacy algorithms in triangle counting: In edge difference privacy, two neighbor graphs differ only by one edge. The addition or deletion of an edge amid any two nodes in the graph has no effect on results of query. Using edge privacy, Nisism et al. [81] proposed a method for calculating triangle counting's smooth sensitivity and minimum spanning tree's cost. An effective approach for generating approximation solutions to subgraph counting questions such as K-triangle counting, triangle counting, and k-star counting is proposed by Karwa et al. [78]. These techniques provide edge privacy and can be viewed as an extension of the algorithm in [78]. The algorithm addresses the broader class of subgraph counting issues with privacy guarantees and improved accuracy. Sun et al. [82] introduced the first method where the count of k-triangle of composite graph is publicized considering edge difference privacy. This technique enabled the published composite graphs to handle any query regarding triangle counting with any constant k.

Qian et al. [83] proposed that all the features of nodes and graphs cannot be represented completely using the degree distribution concept for social network graphs. The technique of publicizing the histogram for computing the node strength is used for edge differential privacy. To optimize the accuracy of publication, two sequence and density-based bucket clustering methods are proposed. In this method, t-bound graphs are constructed to keep the size of the edge weight to a minimum.

Node privacy algorithms in triangle counting: Because node privacy provides a sturdy privacy assurance, a significant quantity of noise must be supplied, resulting in a drastic deformation of the graph structure and poor usefulness. The generic drop to privacy over a graph with bounded degree is one of the most extensively used methods. If a graph with highest degree of d is considered, removing or inserting a node may have the greatest effect on $(d/2)$ triangles. High-degree nodes in graphs with a highest degree larger than d can be eliminated so that the resultant graphs have their highest degree lesser than the threshold. This bounded degree

graph's number of triangles can be a good estimate to the true question response. As a matter of fact, networks with a few high-degree nodes can use this method in order to achieve node privacy for triangle counting.

Zhang et al. [80] presented a method which utilizes degree ordering for edge removal. In the context of node, it addresses the differential privacy issue of increasing sensitivity of the node degree distribution. Under node difference privacy, two histogram techniques of degree distribution are provided: SER-cumulative histogram and SER histogram. For privacy of social network graphs, two types of uncertain graph privacy protection algorithms are proposed by Wu et al. [84]. For privacy protection using uncertain graph technique, the deterministic graph is converted to probability graph. In social network scenarios where the privacy protection is at utmost priority, the uncertain edge probability assignment algorithm is suitable. However, its data availability must be enhanced. There are two histograms of triangle counting distributions of node differential privacy: cumulative distribution histogram and triangle counting distribution histogram. Although the projection method minimizes query sensitivity, but data processing results in a significant loss of existing graph information. Also, data availability is remarkably low.

Triangle counting protection for nodes has been explored, but the triangle counting protection for edges remains unstudied. Furthermore, the projection algorithm of node triangle counting results in significant information loss of the existing graph and limited data availability. Improving the availability of published data while ensuring differential privacy protection is a significant challenge.

7.2.2 *k*-Triangle counting

A *k*-triangle is made up of *k*-triangles that all have the same edge. It is denoted by $f_{k\Delta}(G)$, where G is the input graph. When triangle counting is prolonged to *k*-triangle counting, it becomes more difficult because calculating the smooth sensitivity of *k*-triangle counting is NP-hard. As a result, current approaches primarily emphasize on a trivial value of *k*, though the counting query of $f_{k\Delta}$ is also difficult.

The main idea of [78] is to calculate (ϵ, δ) differential privacy (edge privacy), and for this computation, the noise is added relative to a second-order local sensitivity in place of a "smooth" upper bound. $\mathrm{LS}_{k\Delta}$ denotes the local sensitivity and cannot be used directly with the Laplace mechanism directly. It was demonstrated that LS' is a deterministic function and have the global sensitivity equal to 1, which means it permits the to publish the query with less noise. Zhang et al. [80] presented another approach that uses a function called ladder function. This function is used for counting *k*-triangle having edge privacy.

7.3 Edge weights

Edge weights in SN may echo the communication frequency, the cost of doing business, the familiarity of a relationship, and other factors associated with sensitive information. An intelligence network is a common example, in which edge weights represent the recurrence with which two institutions communicate. Excessive communication may indicate a problem. A commercial trade network is another example, where edge weights represent the price of a transaction between two businesses.

Liu et al. [85] investigated the issues of conserving the efficacy of statistics of shortest paths among nodes while protecting privacy in edge weights. They proposed two approaches for preserving edge privacy: Gaussian randomization multiplication and greedy perturbation. The greedy perturbation is concerned with the size of the perturbed shortest paths being preserved, whereas the Gaussian randomization is concerned with retaining the same shortest paths before and after perturbation.

Another algorithm called edge weight anonymization for social network analysis is proposed by Das et al. [32]. They created an LP model to safeguard graph properties such as k-nearest neighbors, shortest paths, and minimum spanning trees which can be legitimized as linear edge weight functions. Costea et al. [86] used the Dijkstra algorithm to evaluate shortest paths for evaluating protection quality. Under the assumption that the graph is publicly accessible, it postulated differential privacy algorithms for the protection using the weights of edges. Users can access the graph structure without making any changes, but the edge weights are kept private. To enhance the published data utility and accuracy, an algorithm is proposed by adding Laplace noise to each edge weight by Li et al. [1].

The majority of differentially private algorithms currently in use must make a significant trade-off in utility in order to preserve privacy when analyzing extensive large and multifaceted graph structures. Undeniably, as their primary contributions, several of those techniques strive to ameliorate utility. Furthermore, the intricacy of figuring (smooth) sensitivities increases the complexities of differentially private algorithms, if not NP-hard. Even for the queries of k-triangle counting, the query's structure is NP-hard.

8 Summary

Since social networks are growing very fast in these days, privacy breaches in social networks are of major concern. This chapter discussed a solution to privacy issue of social network, that is, differential privacy. Identity disclosure, attribute disclosure, and link disclosure are major types of disclosure that occur in social network privacy breaches. Then solutions to these concerns are also explained in the chapter. Application of differential privacy for social network analysis is also included.

References

[1] Xiaoye, L., Yang, J., Sun, Z., & Zhang, J. (2017). Differential privacy for edge weights in social networks. *Security and Communication Network*, *2017*, 1–10. doi:https://doi.org/10.1155/2017/4267921

[2] Hsu, T.-S., Liau, C.-J., & Wang, D.-W. (2014). A logical framework for privacy-preserving social network publication. *Journal of Applied Logic*, *12*(2), 151–174.

[3] Kulkarni, A. R., & Yogish, H. K. (2014). Advanced unsupervised anonymization technique in social networks for privacy preservation. *International Journal of Science and Research*, *3*(4), 118–125.

[4] Tripathy, B. K., Sishodia, M. S., Jain, S., & Mitra, A. (2014). Privacy and anonymization in social networks. *Intelligent Systems Reference Library*, *65*, 243–270.

[5] Jiang, H., Pei, J., Yu, D., Yu, J., Gong, B., & Cheng, X. Applications of Differential Privacy in Social Network Analysis: A Survey. *IEEE Transactions on Knowledge and Data Engineering, pre-print available at: https://www.computer.org/csdl/journal/tk/5555/01/09403974/1sLH8K2Abp6*.

[6] Dalenius, T. (1977). Towards a methodology for statistical disclosure control. *Statistik Tidskrift*, *15*(429-444), 2–1.

[7] Sweeney, L. (2002). k-Anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *10*(05), 557–570.

[8] Machanavajjhala, A., Kifer, D., Gehrke, J., & Venkitasubramaniam, M. (2007). L-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *1*(1), 3–es.

[9] Li, N., Li, T., & Venkatasubramanian, S. (2007). t-Closeness: Privacy beyond k-anonymity and l-diversity. In 2007 IEEE 23rd International Conference on Data Engineering (pp. 106–115). IEEE.

[10] Wong, R. C.-W., Li, J., Fu, A. W.-C., & Wang, K. (2006). (α, k)-Anonymity: An enhanced k-anonymity model for privacy preserving data publishing. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 754–759).

[11] Zheleva, E., & Getoor, L. (2011). Privacy in Social Networks: A Survey. In Aggarwal, C. eds. *Social Network Data Analytics*. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-8462-3_10.

[12] Backstrom, L., Dwork, C., & Kleinberg, J. (2007). Wherefore art thou r3579x: Anonymized social networks, hidden patterns, and struct. steganography. In Proceedings of International World Wide Web Conference. pp. 1–10.

[13] Campan, A., Turta, T. M. (2009). A Clustering Approach for Data and Structural Anonymity. In Proceedings of the 2nd ACM SIGKDD International Workshop on Privacy, Security, and Trust in KDD (PinKDD'08), in Conjunction with KDD'08, Las Vegas, Nevada, USA, pp 33–54.

[14] Hay, M., Miklau, G., Jensen, D., & Towsley, D. (August 2008). Resisting structural identification in anonymized social networks. In Proceedings of the VLDB EndowmentVolume 1Issue 1, pp 102–114.

[15] Hay, M., Miklau, G., Jensen, D., Weis, P., & Srivastava, S. Anonymizing social networks. Technical report, University of Massachusetts, Amherst, March 2007.

[16] Korolova, A., Kenthapadi, K., Mishra, N., & Ntoulas, A. (2009). Releasing search queries and clicks privately. In International World Wide Web Conference Committee (IW3C2), pp 171–180.

[17] Liu, K., & Terzi, E. (2008). Towards identity anonymization on graphs. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data, Pages 93–106.

[18] Narayanan, A., & Shmatikov, V. (2009). De-anonymizing social networks. In *30th IEEE Symposium on Security and Privacy*, 2009, pp. 173–187.

[19] Wondracek, G., Holz, T., Kirda, E., & Kruegel, C. (2010). A practical attack to de-anonymize social network users. In IEEE *Symposium on Security and Privacy*, pp. 223–238.
[20] Ying, X., & Wu, X. (2008). Randomizing social networks: A spectrum preserving approach. In Proceedings of the SIAM International Conference on Data Mining, pp.739–750.
[21] Zhou, B., & Pei, J. (2008). Preserving privacy in social networks against neighbourhood attacks. In Proceedings of IEEE 24th International Conference on Data Engineering, pp. 506–515.
[22] Zou, L., Chen, L., & Ozsu, M. T. (2008). K-Automorphism: A general framework for privacy preserving network publication. In Proceedings of the VLDB Endowment. Vol. 2 No.1, pp 946–957.
[23] Sweeney, L. (2002). Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty*, *10*(5), 571–588.
[24] Narayanan, A., & Shmatikov, V. (2008). Robust de-anonymization of large sparse datasets. *Security and Privacy* (pp. 111–125).
[25] Lindamood, J., Heatherly, R., Kantarcioglu, M., & Thuraisingham, B. (2009). Inferring private information using social network data. In Proceedings of the 18th international conference on World wide web, Pages 1145–1146.
[26] Zheleva, E., & Getoor, L. (2009). To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles. In Proceedings of the 18th international conference on World wide web, Pages 531–540.
[27] Sihag, V. K. (2012). A clustering approach for structural k-anonymity in social networks using genetic algorithm. In Proceedings of the CUBE International Information Technology Conference. pp. 701–706.
[28] Backstrom, L., Dwork, C., & Kleinberg, J. (2011). Wherefore art thou r3579x: Anonymized social networks, hidden patterns, and struct. steganography. In Communications of the ACM, Vol. 54, Issue 12, pp 133–141.
[29] Bhagat, S., Cormode, G., Krishnamurthy, B., & Srivastava, D. (2009). Class-based graph anonymization for social network data. In Proceedings of the VLDB Endowment, Volume 2 Issue 1, pp 766–777.
[30] Korolova, A., Motwani, R., Nabar, S. U., & Xu, Y. (2008). Link privacy in social networks. In Proceedings of the 17th ACM conference on Information and knowledge management, Pages 289–298.
[31] Zheleva, E., & Getoor, L. (2007). Preserving the privacy of sensitive relationships in graph data. *PinKDD* (pp. 153–171).
[32] Das, S., Egecioglu, E., & Abbadi, A. E. (2010). Anonymizing weighted social network graphs. In IEEE 26th International Conference on Data Engineering (ICDE 2010), pp. 904–907.
[33] Liu, L., Wang, J., Liu, J., & Zhang, J. (2009). Privacy preservation in social networks with sensitive edge weights. In Proceedings of the SIAM International Conference on Data Mining, pp. 954–965.
[34] Barbaro, M., & Zeller, T. (2006 Aug)A face is exposed for AOL searcher no. 4417749. *New York Times*.
[35] Cortes, C., Pregibon, D., & Volinsky, C. (2002). Communities of interest. In Intelligent Data Analysis. vol. 6, no. 3, pp. 211–219.
[36] Choffnes, D. R., Duch, J., Malmgren, D., Guimera, R., Bustamante, F. E., & Amaral, L. (2009 Jun). Swarmscreen: Privacy through plausible deniability in p2p systems tech. Technical Report NWU-EECS-09-04, Department of EECS, Northwestern University.
[37] Cormode, G., Srivastava, D., Yu, T., & Zhang, Q. (2008). Anonymizing bipartite graph data using safe groupings. In Proceedings of the VLDB Endowment, Volume 1 Issue 1, pp 833–844.
[38] Barth-Jones, D. C. (2012 Jul). The 'Re-Identification' of Governor William Weld's Medical Information: A Critical Re-Examination of Health Data Identification Risks and Privacy

Protections, Then and Now. Tech. rep. Columbia University – Mailman School of Public Health, Department of Epidemiology.
[39] Samarati, P., & Sweeney, L. (1998). Protecting Privacy when Disclosing Information: K-Anonymity and Its Enforcement through Generalization and Suppression. Tech. rep.
[40] Machanavajjhala, A., et al (2006). l-Diversity: Privacy Beyond k-Anonymity. In Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006 3–8 April 2006 (p. 24). Atlanta, GA, USA.
[41] Li, N., Li, T., & Venkatasubramanian, S. (2007). t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In: 2007 IEEE 23rd International Conference on Data Engineering (pp. 106–115).
[42] Dwork, C. (2008). Differential privacy: A survey of results. In Agrawal, M., Du, D., Duan, Z., & Li, A. Eds. *Theory and applications of models of computation, ser. lecture notes in computer science* (pp. 1–19). Springer, Berlin/ Heidelberg.
[43] Kifer, D., & Machanavajjhala, A. (2011). No free lunch in data privacy. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (pp. 193–204).
[44] Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In Theory of cryptography conference (pp. 265–284). Springer.
[45] Hay, M., Li, C., Miklau, G., & Jensen, D. (2009). Accurate estimation of the degree distribution of private networks,. In 2009 Ninth IEEE International Conference on Data Mining (pp. 169–178). IEEE.
[46] Task, C., & Clifton, C. (2012). A guide to differential privacy theory in social network analysis. In 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (pp. 411–417). IEEE.
[47] Kossinets, G., & Watts, D. J. (2006). Empirical analysis of an evolving social network. *science*, *311*(5757), 88–90.
[48] Task, C., & Clifton, C. (2014). What should we protect? defining differential privacy for social network analysis. In *State of the art applications of social network analysis* (pp. 139–161). Springer.
[49] Abdulhamid, S. M., Ahmad, S., Waziri, V. O., & Jibril, F. N. (2014). Privacy and national security issues in social networks: The challenges. arXiv preprint arXiv:1402.3301.
[50] Dey, R., Tang, C., Ross, K., & Saxena, N. (2012) Estimating age privacy leakage in online social networks. In 2012 proceedings IEEE infocom (pp. 2836–2840). IEEE.
[51] Kosinski, M., Stillwell, D., & Graepel, T. (2013). Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, *110*(15), 5802–5805.
[52] Gong, N. Z., Talwalkar, A., Mackey, L., Huang, L., Shin, E. C. R., Stefanov, E., Shi, E., & Song, D. (2014). Joint link prediction and attribute inference using a social-attribute network. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *5*(2), 1–20.
[53] Ji, S., Li, W., Gong, N. Z., Mittal, P., & Beyah, R. A. (2015). On your social network de-anonymizablity: Quantification and large scale evaluation with seed knowledge. In *NDSS*.
[54] Ji, S., Li, W., Srivatsa, M., & Beyah, R. (2014). Structural data deanonymization: Quantification, practice, and implications. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (pp. 1040–1053). ACM.
[55] Qian, J., Li, X.-Y., Zhang, C., & Chen, L. (2016). De-anonymizing social networks and inferring private attributes using knowledge graphs. In IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications (pp. 1–9). IEEE.
[56] Ji, S., Wang, T., Chen, J., Li, W., Mittal, P., & Beyah, R. (2017). De-sag: On the de-anonymization of structure-attribute graph data. *IEEE Transactions on Dependable and Secure Computing* 16(4), pp. 594–607.

[57] Shirani, F., Garg, S., & Erkip, E. (2018). Optimal active social network de-anonymization using information thresholds. In 2018 IEEE International Symposium on Information Theory (ISIT) (pp. 1445–1449). IEEE.
[58] Shao, Y., Liu, J., Shi, S., Zhang, Y., & Cui, B. (2019). Fast deanonymization of social networks with structural information. *Data Science and Engineering*, *4*(1), 76–92.
[59] Jakobsson, M. (2005). Modeling and preventing phishing attacks. In Financial Cryptography and Data Security. FC 2005 Patrick, A. S., & Yung, M. Eds. *Lecture Notes in Computer Science* vol. 3570. Springer, Berlin, Heidelberg.
[60] Spear Phishing Attacks. 2017. Retrieved from http://www.microsoft.com/protect/yourself/phishing/spear.mspx.
[61] Gupta, P., Gottipati, S., Jiang, J., & Gao, D. (2013). Your love is public now: Questioning the use of personal information in authentication. In Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, pp. 49–60.
[62] Afroz, S., Caliskan-Islam, A., Stolerman, A., Greenstadt, R., & McCoy, D. (2014). Doppelganger finder: Taking stylometry to the underground. In IEEE Symposium on Security and Privacy (pp. 212–226). San Jose, CA.
[63] Bartunov, S., Korshunov, A., Park, S.-T., Ryu, W., & Lee, H. (2012). Joint link-attribute user identity resolution in online social networks. In Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining, Workshop on Social Network Mining and Analysis. ACM pp. 1–9.
[64] Goga, O., Lei, H., Parthasarathi, S. H. K., Friedland, G., Sommer, R., & Teixeira, R. (2013). Exploiting innocuous activity for correlating users across sites. In Proceedings of the 22nd international conference on World Wide Web, Pages 447–458.
[65] Goga, O., Perito, D., Lei, H., Teixeira, R., & Sommer, R. (2013). Large-scale Correlation of Accounts Across Social Networks. Technical report. International Computer Science Institute. Technical Report TR-13-002, Berkeley, California.
[66] Minkus, T., Ding, Y., Dey, R., & Ross, K. W. (2015). The city privacy attack: Combining social media and public records for detailed profiles of adults and children. In Proceedings of the 2015 ACM on Conference on Online Social Networks, Pages 71–81.
[67] Sweeney, L. (2002). k-Anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10 *5*(2002), 557–570.
[68] Kosinski, M., Stillwell, D., & Graepel, T. (2013). Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, 110 *15*(2013), 5802–5805.
[69] Gong, N. Z., Talwalkar, A., Mackey, L., Huang, L., Shin, E. C. R., Stefanov, E., Shi, E., & Song, D. (2014). Joint link prediction and attribute inference using a social-attribute network. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *5*(2), 1–20.
[70] Labitzke, S., Werling, F., Mittag, J., & Hartenstein, H. (2013). Do online social network friends still threaten my privacy? In Proceedings of the third ACM conference on Data and application security and privacy (pp. 13–24).
[71] Weinsberg, U., Bhagat, S., Ioannidis, S., & Taft, N. (2012). Blurme: Inferring and obfuscating user gender based on ratings. In Proceedings of the sixth ACM conference on Recommender systems (pp. 195–202).
[72] Chaabane, A., Acs, G., Kaafar, M. A. et al. (2012). You are what you like! information leakage through users' interests. In Proceedings of the 19th Annual Network & Distributed System Security Symposium (NDSS). Citeseer.
[73] Kosinski, M., Stillwell, D., & Graepel, T. (2013). Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, *110*(15), 5802–5805.

[74] Backstrom, L., Dwork, C., & Kleinberg, J. (2007). Wherefore art thou R3579X? Anonymized social networks, hidden patterns, and structural steganography. 16th International World Wide Web Conference. Banff, Alberta, Canada.
[75] Yu, H., Gibbons, P., Kaminsky, M., & Xiao, F. (2008). Sybil-Limit: A near-optimal social network defense against sybil attacks. *S&p* (pp. 3–17).
[76] Schonfeld, E. (2008). Techcrunch: Twitter starts blacklisting spammers. http://www.techcrunch.com/2008/05/07/twitter-starts-blacklisting-spammers/.
[77] Raskhodnikova, S., & Smith, A. (2015). Efficient Lipschitz extensions for high dimensional graph statistics and node private degree distributions. arXiv preprint arXiv:1504.07912.
[78] Karwa, V., Raskhodnikova, S., Smith, A., & Yaroslavtsev, G. (2011). Private analysis of graph structure. *Proceedings of the VLDB Endowment*, *4*(11), 1146–1157.
[79] Kasiviswanathan, S. P., Nissim, K., Raskhodnikova, S., & Smith, A. (2013). Analyzing graphs with node differential privacy. In Theory of Cryptography Conference (pp. 457–476). Springer.
[80] Zhang, J., Cormode, G., Procopiuc, C. M., Srivastava, D., & Xiao, X. (2015). Private release of graph statistics using ladder functions. In Proceedings of the 2015 ACM SIGMOD international conference on management of data (pp. 731–745).
[81] Nissim, K., Raskhodnikova, S., & Smith, A. (2007). Smooth sensitivity and sampling in private data analysis. In Proceedings of the thirty-ninth annual ACM symposium on Theory of computing (pp. 75–84).
[82] Sun, H., Xiao, X., Khalil, I., Yang, Y., Qin, Z., Wang, H., & Yu, T. (2019). Analyzing subgraph statistics from extended local views with decentralized differential privacy. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (pp. 703–717).
[83] Qin, Z., Yu, T., Yang, Y., Khalil, I., Xiao, X., & Ren, K. (2017). Generating synthetic decentralized social graphs with local differential privacy. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 425–438).
[84] Wu, D., Zhang, B., Jing, T., Tang, Y., & Cheng, X. (2016). Robust compressive data gathering in wireless sensor networks. *IEEE Transactions on Wireless Communications*, *12*(6), 2754–2761.
[85] Liu, L., Wang, J., Liu, J., & Zhang, J. (2008). Privacy preserving in social networks against sensitive edge disclosure. Technical report, Technical Report CMIDA-HiPSCCS (pp. 006–08).
[86] Costea, S., Barbu, M., & Rughinis, R. (2013). Qualitative analysis of differential privacy applied over graph structures. In 2013 11th RoEduNet International Conference (pp. 1–4). IEEE.

Abdul Rahman, Krishnadas Nanath*

Cracking Captcha using machine learning algorithms: an intersection of Captcha categories and ML algorithms

Abstract: Captcha (Completely Automated Public Turing test to tell Computers and Humans Apart) is a challenge-based response test in order to differentiate between a human and a bot. Captcha came into use with the advent of spambots taking up space posing as humans. Captcha took the Internet by storm as it had multiple uses and capabilities like averting comment spam in blogs, safeguarding website registrations, shielding e-mail addresses from scrapers, preventing dictionary attacks, and counteracting search engine bots. This chapter aims at categorizing text Captcha into various types based on inputs from the literature and visual appearance. It then uses a series of machine learning (ML) algorithms to crack the actual Captcha content using training data. The research investigates the cross section of Captcha type and ML algorithm. A dataset of 1,024 Captcha images was considered for conducting this experiment. It identifies which ML algorithm is most effective in cracking Captcha across various categories. In turn, it helps identify the loopholes in the easy identification of Captcha via automated algorithms. This will enable Captcha users to use the most suitable category of Captcha that is least vulnerable to ML crackdown.

Keywords: Captcha, machine learning, artificial intelligence, text analytics

1 Introduction

Nowadays, machine learning (ML) is used in several domains like image recognition, speech recognition, medical diagnosis, and learning associations. The capabilities of ML are vast and versatile in nature, which makes it a two-sided sword as black-hat hackers, and people with malicious intent can use it in a harmful way. One of the common misuses of ML is to crack several security mechanisms that protect consumer data in this digital world.

Captcha (Completely Automated Public Turing test to tell Computers and Humans apart) is a challenge-based reaction test to separate between a human and a bot. Captcha came into utilization with the dawn of spambots occupying memory in organization databases replicating humans. It overwhelmed the web as it could

*Corresponding author: **Krishnadas Nanath,** Middlesex University, Dubai,
e-mail: username.krishna@gmail.com
Abdul Rahman, Middlesex University, Dubai

https://doi.org/10.1515/9783110766745-002

solve several problems like averting comment spam in blogs, safeguarding website registrations, protecting e-mail addresses from scrapers, preventing dictionary attacks, and balancing search engine bots.

With the growth of Captcha and its applications, reCaptcha was introduced. It is an evolved Captcha-based system and is a combination of advanced Turing tests along with browser data testing (cookies). The applications of reCaptcha are similar to a Captcha, and its core purpose is to differentiate between a human and a bot. Captcha faced some criticism, as it was deemed easy to trounce using software developed by agencies. Thus, reCaptcha was introduced with higher security requirements in an attempt to make it difficult to crack it.

With the growth of advanced technologies and stores becoming cheaper, it has become easier for the Captcha to be trounced. The time required to trounce is reduced over the years due to major improvements in ML capabilities. Artificial intelligence (AI), reduced cost of the cloud, reduced cost of hardware, and outsourcing of the ML engines are several factors that have contributed to the time reduction. This chapter is an attempt to understand the power of ML algorithms in breaking down Captcha and understand which types of CAPCTHAs are more prone to break down with ML algorithms. Further, it also highlights the loopholes, privacy, and security issues with RECaptcha. Secondary data of Captcha images are used for this research, and it is under the Creative Commons license to review the information and share it.

2 Literature review

Captcha is a challenge-based response test in order to differentiate between a human and a bot. Captcha came into use with the advent of Spambots taking up space and posing as humans. ReCaptcha was further introduced as evolved Captcha mechanism with better synchronization and advanced features. Captcha and ReCaptcha faced a lot of criticism over the years for varying reasons ranging from privacy to redundancy and other issues.

This section aims to highlight and document the past efforts in the context of Captcha and ReCaptcha. It also documents how advancement in ML capabilities impacted this domain. The literature review proceeds as follows: the methodology of the literature review is presented first. It highlights the process of article collection and its specifics, including types of articles taken into consideration. Further, the research timeline is discussed, and it aims to showcase how research on Captcha has evolved over time based on the advancement of related technologies. The core review is then presented, followed by the identification of gaps. A summary of the literature review process is presented in Figure 1.

Figure 1: Literature review process.

The articles, books, and journals examined throughout the literature review were collected from various scholarly databases and search engines, including Google scholar, Institute of Electrical and Electronics Engineers, and other databases. The collection of the review also includes a few peer-reviewed blogs and magazines discussing how-to articles elaborating captcha issues for people with disabilities. The diversification in the search methods was adopted in order to present a more unified research output while diminishing any chances for institutional bias toward the subject. In order to reduce further bias from the review, all aspects, including the benefits and issues of Captcha and ReCaptcha, were taken into consideration.

The content analysis of initial articles resulted in keywords that could be used for review searches. The following keywords were used: captcha vulnerabilities, Captcha, benefits of Captcha, usage of ReCaptcha, convolutional neural networks (CNN), Recaptcha, neural networks and Captcha, issues with Captcha, and Completely Automated Public Turing test. The search with these keywords resulted in around 67 related articles and papers; these papers were filtered by disregarding the searches which were too basic in nature (included basic introductions and covered simple case studies). This was preceded by removing the reports which were not related to the field and turned up due to the similarity of the keywords. The final collection ensued into a collection of 54 articles relevant to the literature review. A split of publication types is provided in Figure 2.

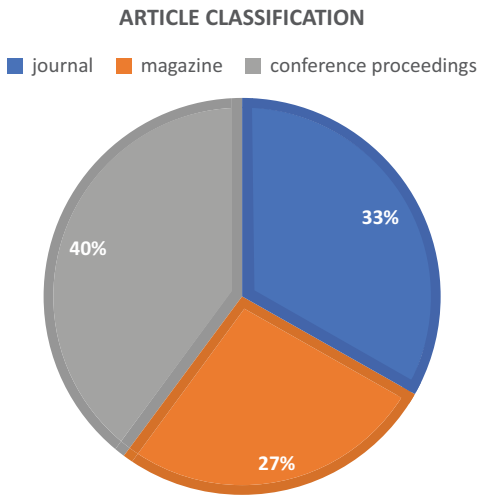


Figure 2: Article classification in the literature review.

These articles were then analyzed over the years to understand the importance given to this field of research. A summary of this timeline is presented in Figure 3. It can be observed that the trend has been increasing over the years, particularly after 2015. With the growth of ML capabilities and computational power, the issues related to Captcha cracking became an interesting area of research. Many articles post-2015 use research and computing techniques to solve the issues arising in this domain.

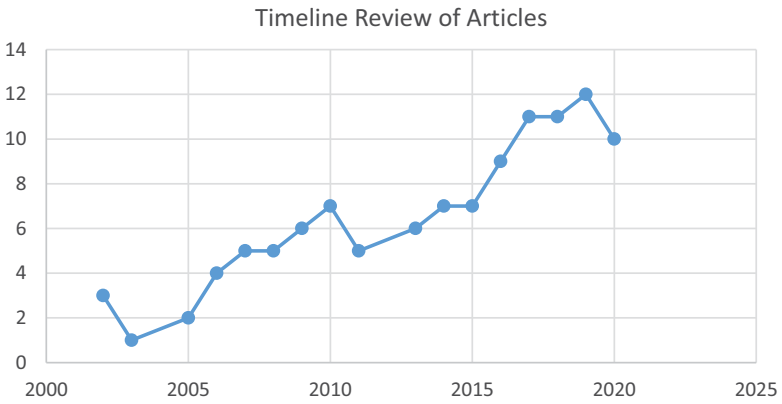


Figure 3: Number of articles published per year.

The relevant research papers started a few years after the introduction of Captcha, as its adoption became widespread, and the number of users grew (Robinson, 2002a). Researchers started documenting issues while finding ways to break Captcha using text-based synthetic analysis techniques and various other methodologies. Some researchers tried to highlight the issues with Captcha from a community perspective. This resulted in the rise of research articles trying to resolve the problems. The widespread use of the Internet and an increasing number of websites (Yahoo, Microsoft, and others) further added to the scope of conducting research. Captcha was used to filter out bots [1], but companies and agents found a commercialized way to break Captcha (a lengthy process), and it became widespread. This was followed by the release of ReCaptcha, where the number of researches reduced, with students and researchers trying out new things with no particular breakthrough.

The year 2016 resulted in an increase in research toward Captcha and ReCaptcha due to evolving technologies in the AI and ML sector. In 2016, major cloud ML platforms (Amazon Web Services and Google cloud) took the Internet by storm. This resulted in people with crunched resources being able to get their hands on advanced technologies with the usage of the Internet. This resulted in the rise of research articles as well.

In order to review the articles, all the papers were examined using a breakdown of the core concepts being discussed in the articles. This resulted in four categories of themes discussed in these papers. The four different themes of articles in the context of Captcha, ReCaptcha, and ML were: theory-driven, critical view of Captcha, supportive view of Captcha, and other related articles.

3 Category 1: theory-driven articles

This category contributed to the maximum number of articles in the selected set; the majority of the work in this category was focused on providing theoretical information. The theoretical information in question is based on various topics such as user experience, the ethics of Captcha and ReCaptcha, the strengths and weaknesses of Captcha, and its reach. Table 1 documents various papers highlighted in this category.

Table 1: Theory-driven articles.

Paper	Description
[2]	This documented research is about the analysis of various ways machine learning can be used to perform optical character recognition. One of the techniques discussed was how an algorithm can be used to break down Captcha.
[3]	This paper is about the analysis of Captcha and how the technology has evolved throughout the years to counter normal Captcha. It also introduced a new type of Captcha using not just characters but numbers to counter the cracking down of Captcha.
[4]	This research paper analyzes text-based Captcha, and describes the pros and cons of using a text-based Captcha for both the designers/attackers of text-based Captcha.
[5]	This research paper highlights the status of Captchas and how its value has changed throughout the years. It also described the way technology impacted the Internet.
[6]	In this documented research, the given content talks about the analysis conducted on the real-world deployed image Captcha. It also analyzes the strengths and weaknesses. The evaluation of security and attacks is also presented.
[7]	This paper talks about Internet security and the part played by Captcha to keep it safe. It recommends that Captcha improves the user experience by keeping the spambots away and defending against different types of Internet attacks. This paper also showcases different types of Captcha used online.
[8]	This is a generic study on how Captcha is used and the way it impacts the Internet.
Robinson (2002b)	This article highlights the history of Captcha and how it turned from a test for artificial intelligence to protect the Internet as a whole from spambots to scammers.

Table 1 (continued)

Paper	Description
[9]	This article highlights the disadvantages of spambots and how Captcha is helping this issue.
[10]	Other than the importance of Captcha, this paper also highlights the criticism given to Captcha and how it might not be useful for all stakeholders.
[11]	The authors examine different types of Captchas to explore the use of different colors to negatively affect the usability or security of the Captchas.

4 Category 2: critical view of captcha

The second category of articles among the selected research set can be categorized as a critical view of Captcha. Most of the publications in this category highlight and demonstrate techniques that can be used in order to trounce Captcha. Recently, there has been a constant increase in the number of articles in this category due to the advancement of technology and ease of access to these technologies. ML and AI have played a critical role in breaking through the wall of this domain. A summary of a few critical papers is presented in Table 2.

Table 2: Critical view of papers.

Paper	Description
[12]	This paper does an analysis on Captcha and the ease of cracking it down using the optical character recognition program (with almost 100% success rate).
[13]	This research highlights the attacks that were carried out on the Asirra Captcha [decom] using machine learning and how it defended against those attacks. The paper also reviews Asirra Captcha.
[14]	This paper introduces a new character segmentation technique of general value that can be used to attack a wide number of text-based Captcha. It demonstrates how easy it is to trounce the text recognition task given by Microsoft Captcha.
[15]	This research is based on how to crack down text-based Captcha using sparse convolutional neural networks. Since many web service providers still use text-based Captcha, this research exposes the loophole with AI algorithms.
[16]	This article illustrates the common method of breaking down Captchas using segmentation.
[17]	This article highlights the ease of cracking down text-based Captchas using automatic segmentation. It also uses the recognition of Captchas with variable orientation and random collapse of overlapped characters.

Table 2 (continued)

Paper	Description
[18]	This paper talks about the robustness of text-based Captchas and how easy it is to crack down on text-based Captchas. It also recommends other alternatives to text-based Captcha.
[19]	This paper talks about machine learning techniques in cracking down text-based Captchas and image-based Captcha. It conducts an analysis on text-based Captcha and image-based Captcha.
[20]	In this paper, the authors suggest cost-effective techniques to find loopholes in the Captcha system.

These papers provided a new perspective on working toward solutions that could improve the domain of Captcha. The loopholes and vulnerabilities were exposed by several researchers, and this provided a platform to make the system more efficient.

5 Category 3: supportive view of captcha

This set of papers supports the use of Captcha and expresses the need for using it in the long run. However, these papers were mostly published before 2004, and hence one should be careful when deriving insights, although a few articles suggest ways to improvise and keep up with the trend, making it notable for the review. A summary of these papers is presented in Table 3.

Table 3: Supportive view of captcha.

Paper	Description
[21]	This research describes the way one can use Captcha to safeguard from the dangers of the Internet (spambots, data security, and others).
[22]	This paper outlines various ways to stop the illegal cracking of software on mobile phones through the use of Captcha. Since phones are not able to process optical character recognition programs, it could help in cracking Captcha.
Robinson (2002a)	This article does a study on how Captcha helped in distinguishing between humans and computers. It also describes how Yahoo kept out rogue spammers from its database.
[23]	This article mentions that there is no method that can guarantee spam protection completely. Internet users will always end up finding different ways to attack using spam, no matter how strong the defense mechanism is. Hence, Captcha falls in the same category and has its positives and negatives.

Table 3 (continued)

Paper	Description
[24]	This article illustrates how e-commerce uses a Captcha to defend itself from attacks that are possible through spambots and also does an analysis on different types of Captchas.
[25]	This article highlights how Captcha has been effective in getting rid of spammers.

6 Category 4: other articles

There were few articles that could not be categorized into the parts above and hence have been put in this category. A summary of articles is presented in Table 4.

Table 4: Other articles.

Paper	Description
[26]	This article is based on how Captcha came crashing down on those who are blind or visually impaired. This research raised questions on inclusivity.
Salvatore (2018)	This article talks about the various types of Captchas and mentions how easy it is to break down Captcha using deep learning, even if one is a rookie.

The review of these articles in all categories overall illustrates how both captcha and ML have evolved throughout the years. This review reveals that there is a need for an alternate Turing system mechanism. The current technologies (like Re-Captcha) can be considered safe to use but come with a lot of privacy and ethical issues. Therefore, a better alternative is needed for making this domain sustainable for further applications.

The review also suggests that Captcha and ReCaptcha have not failed, but the advancement of ML technologies is the reason for their downfall. The downfall of Captcha and ReCaptcha is inversely related to the advancement of ML capabilities and ease of its access. Other than the growth of computational power and ML, there are inclusivity issues also that Captcha faces. Thus, a better alternative can be crafted if Captcha is proven desolate.

7 Research method

The core idea of the experiment was to propose various categories of Captcha that have not been explored in previous research papers. The dataset of Captcha images was divided into the proposed categories. These categories were then monitored across various ML algorithms to identify those categories which are more vulnerable to get cracked by the techniques. This could provide design guidelines for Captcha designers for avoiding the crackdown from ML algorithms.

It was decided to use a public dataset to build the research design. This could assist in a large set of Captcha images to learn from and develop categories. It could provide ample training data for the successful working of the ML model. While there are versions of the dataset available on Kaggle, this research used a modified and cleaned dataset by Rodrigo Wilhelmy available on ResearchGate [27]. The dataset consisted of 1,040 images along with 1,040 labels for the images. The Captcha dataset is based on the most common captchas used, and the images in the dataset are five-letter words that contain a combination of numbers and alphabets. The five-letter words provide consistency in the application of ML algorithms for cracking and predicting the actual letters and numbers. The bias of the length is removed by maintaining consistency. A sample Captcha and its correct classification in the text is provided in Figure 4.

Captcha sample image	Correct text classification
	2b827

Figure 4: Sample Captcha in the dataset.

The dataset was further categorized into interesting categories that have not been explored in the literature. Since the images were located in one folder with one Captcha in a single image, codes were developed in Python to classify these images into multiple categories. Once the category classification was completed, they were appended to the dataset. The categories are based on the following parameters: crossline, blur percentage, and Captcha boundary. The crossline categories describe the way a crossover line is drawn across the Captcha image. This could be angular or perpendicular in nature. Blur percentage indicates the percentage of Captcha images that are blurred compared to the bold characters. The Captcha boundary indicates the first and last characters of the image. This could be numbers on both ends, characters on both ends, or a mix of numbers and characters. A summary of these categories is presented in Table 5.

Table 5: Categories of Captcha.

Category	Subcategory	Explanation
Crossline category	Angular category	The crossover line on the Captcha image is at an angle (not in parallel) to the direction of the text.
	Perpendicular category	The crossover line on the Captcha image is in parallel to the direction of the text.
Blur percentage	Majority	More than 50% of the image includes blurred characters and numbers.
	Minority	Less than 50% of the image includes blurred characters and numbers.
	Equal	The ratio of blurred characters/numbers to the bold ones is 1.
Captcha boundary	Numbers	The first and the last characters of the Captcha image are numbers.
	Characters	The first and the last characters of the Captcha image are letters.
	Number_Character	The first and the last characters of the Captcha image include both a number and a character.

The codes for the entire process were developed using Python, and the following libraries were used:

a. Matplotlib: It is a cross-platform data analysis and interactive plotting library written in Python for use with NumPy's numerical extension. It represents a feasible alternative to MATLAB in the open-source world. Additionally, developers can use matplotlib's Application Programming Interfaces to integrate plots in graphical user interface software.
b. Numpy: It is a Python module that adds support for huge, multidimensional arrays and matrices, as well as a large set of high-level mathematical functions for working with these arrays.
c. Keras: It is a robust and simple-to-use Python library for designing and testing deep learning models. It wraps the quick numerical computing libraries "Theano" and "TensorFlow" and enables the definition and training of neural network models in a few lines of code. It reduces the amount of user activities needed to complete basic tasks and delivers simple and actionable error messages.
d. TensorFlow: It is a data flow graph-based software library for numerical computation of mathematical expressions. The graph's nodes represent mathematical operations, while the edges correspond to the multidimensional data arrays (tensors) that pass between them

In order to implement the pipeline of ML algorithms that could crack the Captcha images in the dataset, the following steps were used:

1. Identify the unique letter and numbers represented in the images.
2. Create an array of indices and shuffle it if necessary.
3. Determine the sample size for preparation.
4. Divide the data into training and validation sets.
   a. Examine the picture.
   b. Decode and grayscale transform.
   c. Convert to float32 in the range [0, 1].
   d. Resize the image to the correct dimensions.
   e. Transpose the picture such that the time dimension corresponds to the distance of the image.
   f. Convert the characters in the mark to numerical values.
5. Create a directory since our model needs two inputs.
6. Calculate the training time failure value and use self.add loss() to adjust it to the layer.
7. Return the computed predictions after the evaluation.
8. Include the initial convolutional block and then incorporate the second convolutional block.
9. Use two maximum pools, each with a different pool height and strides.
10. Until forwarding the performance to the RNN component of the model, reshape it appropriately.
11. Provide the model with photographs and add an output layer, followed by a Connectionist temporal classification (CTC) layer that will calculate CTC loss at each stage for improved analysis.
12. Set the number of folds (epoch): An epoch is a concept used in ML that refers to the number of passes the ML algorithm has made over the entire testing dataset. Typically, datasets are organized into batches (especially when the amount of data is very large). Certain individuals use the word iteration imprecisely, referring to the process of running one batch through the model as an iteration.
    Each epoch updates the dataset's internal model parameters. As a result, the batch gradient descent learning algorithm is named for a single batch epoch. Typically, an epoch's batch size is one or more and is often an integer value in the epoch series. Determining how many epochs a model can run to learn is dependent on a number of parameters relating to both the data and the model's objective, and although attempts have been made to automate this method, a thorough interpretation of the data is often required.
13. Perform the training and testing procedures.

8 Results and conclusion

The steps were followed, and the results were documented across various categories. The experiments of cracking Captcha were not only performed across various categories but also across various ML algorithms. The best solution was to combine a CNN and a fully connected classifier. This resulted in an overall accuracy of 93%. A summary of execution is presented in Table 6.

Table 6: Results of ML detection.

Category	Subcategory	Percentage of cases not detected
Crossline category	Angular category	1.54
	Perpendicular category	3.85
Blur percentage	Majority	11.54
	Minority	3.85
	Equal	7.69
Captcha boundary	Numbers	6.15
	Characters	6.15
	Number_Character	8.46

As it can be observed that while the overall accuracy was 94%, the insights on category prediction provide additional value. The last column in Table 6 indicated the percentages of cases that could not be cracked by the best ML algorithm. It can be observed that Captcha images with blur percentage in the majority were the toughest to crack. This could provide future design guidelines on Captcha images to prevent against ML crackdown. In the crossline category, it was found that the perpendicular category had more cases of defense against crackdown when compared to the angular category. When the boundary of Captcha is considered (first and the last), it was found that a mix of numbers and alphabet provided the maximum resistance to ML algorithm crackdown. Therefore, if all the categories are combined, the best design of Captcha would have perpendicular crossover, blur space in the majority, and a mix of number and alphabet as the first and last characters of the text.

While this is a preliminary experiment to understand the design, it opens up the conversation around best designs to defend against ML crackdown. The categories were designed based on the available Captcha images in the dataset. However, with a greater variety of test images, more meaningful categories can be designed. Further, the sample size can be increased in the future to test the robustness of the model with various algorithms in place.

List of acronyms

CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
ML	Machine learning
AI	Artificial intelligence

References

[1] Guo, Y., Luo, X., Chen, J., & Zhang, Y. (2017). A survey on breaking technique of text-based captcha. *Security and communication networks*, 2017.

[2] Jain, V., Dubey, A., Gupta, A., & Sharma, S. (2016, March). Comparative analysis of machine learning algorithms in OCR. In 2016 3rd International conference on computing for sustainable global development (INDIACom) (pp. 1089–1092). IEEE.

[3] Gupta, A., Jain, A., Raj, A., & Jain, A. (2009). *Sequenced tagged captcha: Generation and its analysis*. IEEE.

[4] Bursztein, E., Martin, M., & Mitchell, J. (2011). *Text-based captcha strengths and weaknesses*. ACM.

[5] Guo, F., Wang, M., Li, Y., & Yan, H. (2011). *The vulnerabilities and status of captchas*. IEEE.

[6] Zhao, B., Weng, H., Ji, S., Chen, J., Wang, T., He, Q., & Beyah, R. (2018). *Towards evaluating the security of real-world deployed image captchas*. ACM.

[7] Tanthavech, N., & Nimkoompai, A. (2019). *Captcha: Impact of website security on user experience*. ACM.

[8] Jeng, A. B., Tseng, C. C., Tseng, D. F., & Wang, J. C. (2010). A study of Captcha and its application to user authentication. In International Conference on Computational Collective Intelligence (pp. 433–440). Springer, Berlin, Heidelberg.

[9] Garfinkel, S. (2003). Excuse me, are you human? *Technology Review (1998)*, *106*(5), 28.

[10] David, L. M. (2006). Human or Computer? *InfoWorld*, *28*(24), 17.

[11] El Ahmad, A. S., Yan, J., & Ng, W.-Y. (2012). Captcha Design: Color, Usability, and Security. *IEEE Internet Computing*, *16*(2), 44–51. doi:10.1109/MIC.2011.102

[12] Yan, J., & El Ahmad, A. S. (2007). *Breaking visual captchas with naive pattern recognition algorithms*. IEEE.

[13] Golle, P. (2008). *Machine learning attacks against the Asirra Captcha*. ACM.

[14] Yan, J., & El Ahmad, A. (2008). *A low-cost attack on a Microsoft captcha*. ACM.

[15] Ferreira, D. D., Leira, L., Mihaylova, P., & Georgieva, P. (2019). *Breaking text-based captcha with sparse convolutional neural networks*. Cham: Springer International Publishing.

[16] Haselsberger, J. (2007). Captcha gotcha. *Science News (Washington)*, *171*(20), 319.

[17] Starostenko, O., Cruz-Perez, C., Uceda-Ponga, F., & Alarcon-Aquino, V. (2015). Breaking text-based captchas with variable word and character orientation. *Pattern Recognition*, *48*(4), 1101–1112. doi:10.1016/j.patcog.2014.09.006

[18] Gao, H., Wang, X., Cao, F., Zhang, Z., Lei, L., Qi, J., & Liu, X. (2016). Robustness of text-based completely automated public Turing test to tell computers and humans apart. *IET Information Security*, *10*(1), 45–52. doi:10.1049/iet-ifs.2014.0381

[19] Tang, M., Gao, H., Zhang, Y., Liu, Y., Zhang, P., & Wang, P. (2018). Research on deep learning techniques in breaking text-based captchas and designing image-based captcha. *IEEE Transactions on Information Forensics and Security*, *13*(10), 2522–2537. doi:10.1109/TIFS.2018.2821096

[20] Yu, N., & Darling, K. (2019). A low-cost approach to crack python captchas using AI-based chosen-plaintext attack. *Applied Sciences*, *9*(10), 2010. doi:10.3390/app9102010
[21] Byers, S., Rubin, A., & Kormann, D. (2002) *Defending against an Internet-based attack on the physical world* ACM.
[22] Shirali-Shahreza, M., & Shirali-Shahreza, S. (2006). *Preventing mobile software cracking using captcha*. IEEE.
[23] Robb, D. (2005). Spam Bashing: The 98 percent solution. *Business Communications Review*, *35*(3), 46.
[24] Pope, C., & Kaur, K. (2005). Is it human or computer? Defending e-commerce with captchas. *IT Professional*, *7*(2), 43–49. doi:10.1109/MITP.2005.37
[25] Kesmodel, D. (2006). Codes on sites 'captcha' anger of web users. *The Wall Street journal. Eastern Edition* B1–B2 page.
[26] Spangler, T. (2007). As Web evolves, blind left behind. *Chicago Defender (1973)*, *101*(158).
[27] Wilhemy, R. (2021) https://www.researchgate.net/publication/248380891_captcha_dataset
[28] Robinson, S. (2002a). Captcha puzzles help unmask computers acting as people; Yahoo targets keeping out rogue spammers academics join in difficult development: Ontario Edition. In *Toronto star*.
[29] Robinson, S. (2002b). Human or computer? Take this test. *The New York Times*, *152*, D1.

Kiran Aswal, Dinesh C. Dobhal*, Umesh K. Tiwari, Heman Pathak

The ransomware: an emerging security challenge to the cyberspace

Abstract: Cyberspace is an interactive virtual environment that is created by globally interconnected computing devices, communication technologies, and users. It provides a platform to the users where they can interact, discuss a topic, play games, and create intuitive media, among many other activities. At present time, almost all organizations, institutions, government agencies, and individual users which are connected to the Internet, are major participants of cyberspace. These participants do have critical information that is either stored in their systems or on remote storage systems. These systems are required to be updated but may have some vulnerabilities which makes them a soft target to the cyberattack.

Ransomware, *a continuously evolving second-generation hybrid malware*, is the most preferred and effective tool that is used by cyber-criminals to intrude into such targeted systems. Owing to the complex structure, the ability to escape from a detection system, the damage caused, and rapid growth in the number of affected systems, they are successful in becoming a serious threat to cyberspace.

However, before moving toward designing a practical solution, there is a need to explore and analyze the static and dynamic characteristics of these malicious codes to make aware to researchers and academicians. To fill this gap, in this chapter, we present an evolution of ransomware, their life cycle, concealment techniques used by them, and the machine learning-based approach to defending their attack. We also present a static analysis of *WannaCry ransomware* that could be more useful in designing a defense line against ransomware attacks.

Keywords: cyberspace, cyberattack, ransomware, machine learning, static analysis, Wannacry

***Corresponding author: Dinesh C. Dobhal,** Department of Computer Application, Graphic Era Deemed to be University, Dehradun, Uttarakhand, India, e-mail: dineshdobhal@gmail.com
Kiran Aswal, Heman Pathak, Department of Computer Science, Gurukul Kangri Viswavidyalaya, Dehradun Campus, Uttarakhand, India, e-mail: kiranaswal1984@gmail.com, hpathak@gkv.ac.in
Umesh K. Tiwari, Department of Computer Applications, Graphic Era Deemed to be University, Dehradun, Uttarakhand, India, e-mail: umeshtiwari22@gmail.com

https://doi.org/10.1515/9783110766745-003

1 Introduction

Cyberspace, or Cyberworld, is an interactive virtual environment that is created by globally interconnected computing devices, communication technologies, and users. It provides a platform to the users where they can interact, discuss a topic, play online games, create intuitive media, among many other activities. The term *Cyberspace* was first introduced by William Gibson, *an American-Canadian speculative fiction writer*, in 1982 and defined it as a consensual hallucination that is experienced daily by billions of human beings [1].

At present time, almost all organizations, institutions, government agencies, and individual users which are connected to the Internet are major participants of cyberspace. These participants do have critical information that is either stored either on their local systems or the cloud storage. These systems must have genuine software from reliable sources and should be updated on regular basis. Failing which, these systems may be left with some vulnerabilities, which makes them a soft target for a cyberattack.

A cyberattack is an offensive action that is initiated, *from a remote system*, against a specific computer or a network of computers. These attacks may have different objectives, such as in one case, the objective of the attack could be to disable the availability of the services that are provided by the targeted system, whereas in the second case it could be to gain access to the crucial data. In order to become successful, cyber-criminals use all possible means, *including social engineering*, to hit the targeted systems. Traditional threats were simple, but modern threats are a combination of two or more types of attacks, which are briefly introduced in Table 1.

Table 1: Types of cyberattack.

Type of attack	Description
Malware	This is an attempt to compromise the confidentiality, integrity, or availability of a computer system's resources by designing and disseminating a malicious code. The software code could be in the form of self-executables, interpreted code, scripting code, macros, and so on.
Phishing	In this attack, the attacker misleads the victim into opening an email, or a link to a malicious web page text message by impersonating a trusted and reliable entity. As the victim opens the infected email or the web link, the malicious code is downloaded to his/her computer.
Man in the middle	It is a type of eavesdropping attack, where the attacker can secretly read, insert, and modify the data by intercepting the communication link between the client and the server.
Denial of service	This attack shuts down or crashes a computer system or a network by flooding it with artificial traffic, which makes the services of the targeted system inaccessible to its legitimate users.

Table 1 (continued)

Type of attack	Description
SQL injection	In this attack, the attacker attempts to insert an SQL query into the input data submitted from client to server application. On execution, such SQL queries allow the attacker to gain unauthorized access to sensitive data on the online database server.
Eavesdropping	In this type of attack, the attacker uses an insecure communication network to get unauthorized access to data as it is being sent or received by the user.
Cross-site scripting	This is a code injection attack, in which a malicious script is injected into trusted websites or web applications. On a visit to such websites or applications, the malicious script is executed by the web browser of the victim's system.
Replay	This is an attack, where the attacker delays, replays, or retransmits the valid transmission of the data, between a genuine user and a site, by monitoring the network activities and gaining unauthorized access to the communication link.

Malware, a malicious software, is an unwanted software or scripts that cause a serious dent to the targeted system with their destructive intent. They include but are not limited to viruses, Trojans, backdoors, spyware, or rootkits. The working and the structure of the malware may differ but they do have some common characteristics, which are highlighted in Table 2 [2].

Table 2: Malware characteristics.

Characteristics	Description
Propagation	It describes the method used by malware to propagate and spread itself to other systems. With the advent of a reliable, connected network of computing devices, malware uses it as a preferred medium for propagation. Now it is not limited to floppy disks or USB external disks and can spread very quickly via email, compromised websites, or even directly over the Internet.
Infection	It covers how malware infects the target system, which is usually done by exploiting well-known flaws or configuration errors.
Persistence/ evasion	It explains how the malware avoids detection and remains persistent on the infected computer.
Payload	It specifies the software's actual malicious action, which can range from sending amusing joke messages to erasing the hard disk's contents.

According to the reports [3–5], which are published by well-known research groups, malware is the most preferred vehicle that is used to intrude into the targeted systems. The "Avira Q3, 2020 report" [6] revealed a major increase of up to 50% in the amount of classic malware, exploit-based threats, and coinminer, whereas the "Bitdefender's

Mid-Year Threat Landscape Report 2020" [7] revealed a 715% increase in ransomware attacks on a year-over-year basis. Therefore, the main focus of this chapter is on malware, *particularly ransomware*, its life cycle, the way it is used to intrude into and infect the system. The interested readers may refer to [8–10] for other types of cyberattacks.

Based on the infection vector, and the obfuscation techniques used, the malware is classified as the first-generation malware (aka. *traditional malware*) and the second-generation malware (aka. *advanced malware*) [11].

1.1 First-generation malware

The first-generation malware, *or traditional malware*, is the malicious program whose structure does not change during or after the infection. Even though the static architecture of these malicious codes makes it is easier to detect them by using unique signatures, they have a significant impact on the performance of the system, as well as the integrity, and confidentiality of the information. These malwares, *some of which are presented in* Table 3, also form a base for most destructive second-generation malwares.

Table 3: First-generation malwares.

Traditional malware	Characteristics	Some well-known examples
Virus	These are the early days' malicious codes that rely on computer files, and removable media, like floppy disk, CD/DVD, for transportation from one computer to another. The virus encapsulates itself in the files, or boot sector of removable media so that it can be propagated easily with the transfer of files or removable media. When these infected files are opened in a clean computer, the virus installs itself in the system and starts infecting the files.	*Creeper system* (1971), *Rabbit* (1974), *Brain* (1986), "Ping Pong" (1988), *Michelangelo* (1991), *CIH* (1998)
Worm	This malware is similar to the virus by design but unlike a virus, the mechanism it uses to propagate itself is different. In addition to propagating itself using infected removable media, the worm can spread itself from one computer to another by using network resources through which the computers can communicate with each other. This is a resource-consuming iterative process.	*The Morris* (1988), *Father Christmas* (1988), *Melissa* (1999), *ExploreZip* (1999), *ILOVEYOU* (2000)

Table 3 (continued)

Traditional malware	Characteristics	Some well-known examples
Trojan horse	It is the malicious code that embeds itself into a legitimate program, which has a known effect on the system, but the covert effect of embedded malware is kept hidden from the receiver. These malwares are also known by different names such as remote access Trojan (RAT), which are designed and propagated by an attacker to intrude into and gain complete access to the victim's system, and downloader Trojan that downloads additional content onto the infected computer.	***Remote access Trojans:*** *SubSeven*, *Back Orifice*, *Sakula*, *Havex*, *Agent.BTZ/ComRat*, *Dark Comet* ***Trojan downloader:*** *W32/FraudLoad*, *W32/JQCN*, *W32/Bredolab*, *Trojan. Downloader.JPUY*
Adware	This malware, a.k.a *advertisement-supported software*, displays undesirable, potentially malicious advertisements on compromised systems. It may collect browser history to display the most suitable advertisements on the screen or redirect a user's online search request to a web page that contains information about other products.	*DealSpy*, *ElectroLyrics*, *GetSavin*, *Myplaycity*, *Topic Torch*, *ZenDeals*
Hybrid	A hybrid malware is a malicious code that combines characteristics of more than one type of malwares. For example, malware is both a Trojan and a worm.	*Conficker*, *Zeus*, *Waledac*, *Mariposa*, *Grum*, *Kraken*, *Methbot*, *Kelihos*

1.2 Second-generation malware

Today, the majority of malwares are hybrid by structure and most of them use a combination of Trojan and a worm where they seem to be a Trojan, but after the infection, they behave like a worm. The structure of these malicious programs is dynamic and changes frequently by using the most advanced obfuscation techniques. They can elude a malware detection system. They are further categorized as encrypted, oligomorphic, polymorphic, and metamorphic malware based on the obfuscation technique used to hide their signature.

– *Encrypted malware*: Encryption is one of the obfuscation methods that is used by most of the malwares to evade the traditional malware detection and static code analysis technique. The encrypted malwares may include an encrypted malicious payload and a decryptor program (Figure 1). Once loaded into the system, it scans the system and the decryptor XOR the key and the encrypted malicious payload of the malware to decrypt it. The decrypted malicious code then takes control of the system and performs its destructive intent, and after the infection to the current system, the worm component is activated to propagate the encrypted malware to other systems. However, the decryptor algorithm does not

change, and its signature can be used to analyze and detect the encrypted malware. CASCADE was a prominent encrypted malware of the 1980s for disk operating systems (OS), which infected *.COM files to generate a "falling letter" effect on the screen; later the 32-bit implementation of the same encryption technique was used in the "Zombie/Win95" and "Win95/Mad" malwares [11].

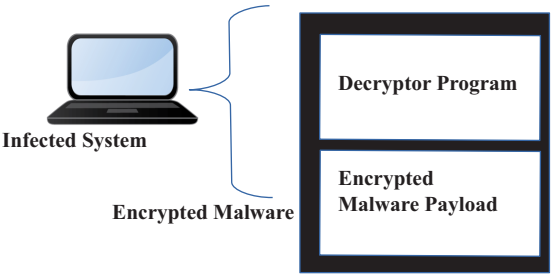


Figure 1: Encrypted malware.

– *Oligomorphic malware*: The decryptor program of encrypted malware does not change its structure and the signature in new variants, which makes it easy for the malware detection system to detect it simply by the signature of the decryptor. This led to the development of oligomorphic malwares (Figure 2), where its structure is similar to the encrypted malware, but its new variant includes a muted decryptor. The oligomorphic malware can have a finite number of slightly different decryptor programs, such as Win95/Memorial malware can generate 96 muted decryptors. Other early examples of oligomorphic malwares are "WordSwap" and "BadBoy (a.k.a *Virus: WM/Badboy.B*)." The traditional malware detection system can detect these malwares if it is updated with the signature of all decryptors. However, it is found that signature-based techniques are not suitable for the detection of oligomorphic malware.

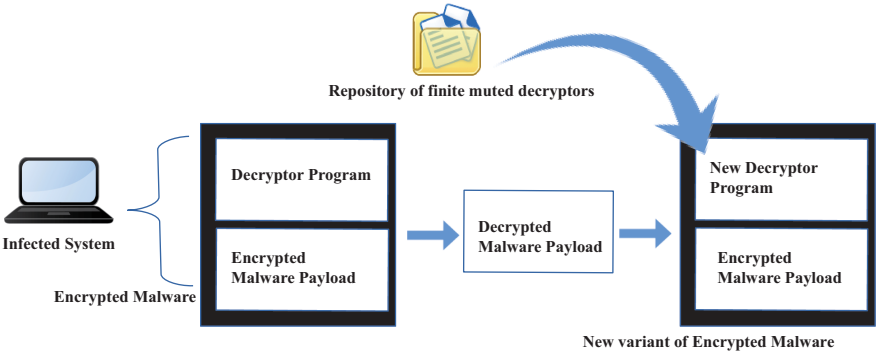


Figure 2: Oligomorphic malware.

– *Polymorphic malware:* Polymorphic malware is similar to oligomorphic malware, but it relies on a mutation engine to generate millions of decryptors every time they infect a system (Figure 3). To get a new variant of polymorphic malware, an encrypted malware payload without mutation is combined with a muted decryptor. For example, 1260 (a.k.a. V2PX), written by Mark Washburn in 1989, was the first polymorphic malware, whereas "URSNIF (a.k.a *Trojan: W32/Ursnif*)," "VIRLOCK (a.k.a *Win32/VirLock Ransomware*)," "VOBFUS (a.k.a *W32/Vobfus; Worm:W32/Vobfus.BK*)," and "BAGLE (a.k.a *Worm:W32/Bagle*)" are few of the most ill-famed polymorphic malwares ever seen. Although a traditional signature-based detection mechanism is not recommended to detect the polymorphic malwares, anomaly-based detection methods are more successful against them.

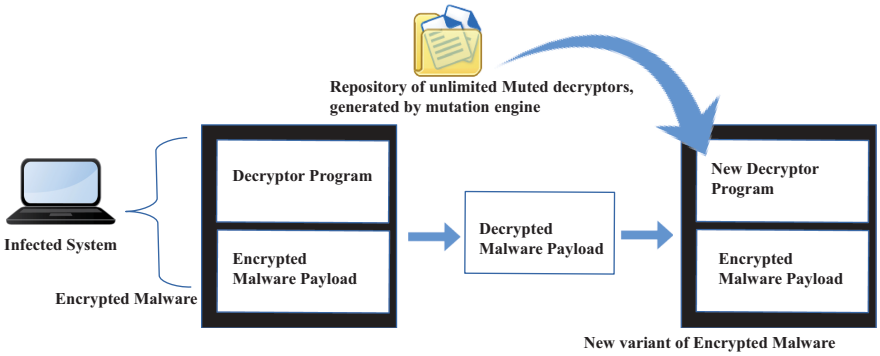


Figure 3: Polymorphic malware.

– *Metamorphic malware*: Unlike malwares, which mutate the decryptor program only, metamorphic malware (Figure 4) mutates itself including the malware payload in such a way that does not change its behavior and intent. This obfuscation technique is also known as body-polymorphic and is almost impossible to be detected by traditional malware detection systems. The malware "Win95/Regswap" was identified as the first metamorphic malware in 1998; thereafter, some of the malwares which possess metamorphic behavior can be named as "Win32/Ghosts," "W32/NGVCK," "PS-MPC."

To explore individual components of malwares, the reader may refer to the online resource maintained by the major antivirus companies and the archives collected and disseminated by CERT (www.cert.org).

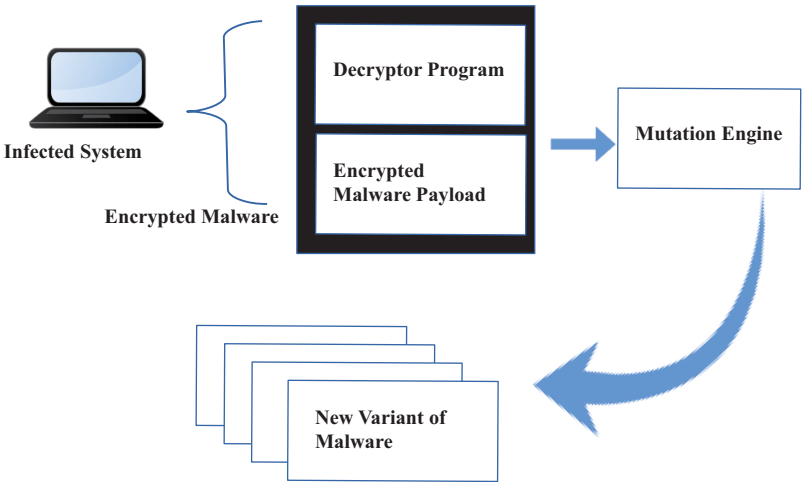


Figure 4: Metamorphic malware.

2 Ransomware: an emerging threat to cyberspace

Owing to the complex structure, the ability to escape from a detection system, the damage caused, and rapid growth in a number of affected systems, ransomware is successful in catching the attention of cybersecurity experts and academicians. The main objective that motivates cyber-criminals to design, create, and spread these malicious and destructive programs is the ransom that is demanded from the targeted organizations or the individual user [12]. However, legitimate users can also contribute to the spread of these malwares by mistake or ignorance that is caused by a lack of awareness.

Ransomwares are second-generation hybrid malwares and most of them are a combination of a Trojan and a worm, which often target the data-sensitive organizations, pharmaceutical industry, hospitals, government agencies, power supply systems, transportation systems, or even the individual user. They accomplish their objective by restricting access to the physical or logical resources of the infected systems and force the user to pay the ransom within the stipulated time. The amount of ransom is smartly decided by the ransomware according to the user's profile and the sensitiveness of the data stored in the system.

Although the notion of ransomware is not new, the popularity of this sort of malicious software has recently risen. In reality, several high-profile ransomware assaults have been reported in recent years. According to the Bitdefender's Report 2020 [7], a significant increase of 715% on a YoY basis in detected and blocked ransomware attacks is highlighted. On November 24, 2014, a large-scale attack was launched against Sony Pictures Entertainment that leaked the confidential data of

the company and threaten to postpone the release of the film *The Interview*. In the line, SamSam (2015), TeslaCrypt (2016), WannaCry (2017), Ryuk (2018), and the Maze ransomware attack on IT giant Cognizant in 2020 can also be mentioned as the worst ransomware attack in the past few years.

The infection vector, payload, and aftereffects make it different from other malwares. There is an ecosystem of cyber-criminals, who developed it as a business model and offer ransomware as a service (RaaS) to other cyber-criminals. The ransomware code can be used simply by a one-time payment, or by sharing a part of the collected ransom. There exists cooperation among the cyber-criminals, where a group is given the responsibility of creation and development of ransomware, and another group initiates a campaign to broadcast the malware, and both of the groups finally enjoy the profit. The majority of ransomwares use social engineering that attracts a user to perform some action, which in turn starts the malware infection phase. This method is more successful against unaware, and less technically sound users. The victims could be convinced that they are identified as involved in some offensive activities, and forcefully redirect them to a compromised web page for automated download of the initial component of the ransomware.

The ransomware can be identified as crypto-ransomware and locker ransomware, where the crypto-ransomwares search the critical user data and files stored on the system and use the most advanced encryption techniques to encrypt them, and the locker ransomwares deny access to the infected system by blocking the input and output interfaces. Although the structure and mode of operation of the ransomware are different, they have some common behavior like data stealing and sending threatening messages. The systems infected by crypto-ransomware function normally, but the systems infected by locker ransomwares provide very limited options to the users. In addition, these ransomwares leave a potential backdoor in the system and spread the infection to other systems. Some of the crypto ransomwares can be named as “CryptoLocker,” “CryptoWall,” and “WannaCrypt,” where “Winlocker” is an example of locker ransomware [13].

These malicious codes are designed, crafted, or propagated for a particular type of target such as general-purpose computers, smartphones, Internet of things (IoT) enabled devices, which run on a specific platform like Windows, Linux, Android, Mac OS, and iOS. According to the statistical data provided by Statcounter Global Stats [14], the worldwide market share of different OSs during February 2020 to February 2021, presented in Table 4, Windows OS is sharing market share of 31.6–37.3%, whereas Android OS is sharing market share of 36.46–41.58%. Therefore, the majority of malwares target Windows and Android-based systems. Since Microsoft Windows share 73.21–77.22% [14] of the desktop OSs market share, from February 2020 to February 2021, it is the most proffered target of cyber-criminals, whereas Android-based systems are next to Windows-based systems. These malwares can be very effective against sensor-based devices, such as wearable devices and the IoT, and can pose a serious challenge to them [15].

Table 4: Worldwide market share of different operating systems.

Platform	Market share (Feb 2020 to Feb 2021)	Market share (Jan 2015 to Dec 2015)
Windows OS	31.6–37.3%	48.63–55.87%
Linux OS	0.7–0.89%	1.46–1.55%*
Android OS	36.46–41.58%	20.46–27.85%
Mac OS	7.01–8.62%	4.82–6.46%
iOS	14.22–17.23%	10.23–11.49%

*Data collected from Statista, 2021 [16].

3 Evolution of the ransomware

Though the first documented ransomware attack was reported in 1989, there was no movement in the growth of ransomware attacks till the mid-2000s. Recently, we have seen a surge in such attacks, where they started using more sophisticated and harder-to-crack obfuscation techniques. Although owing to legal and technical issues, most of the information about the ransomware is either missing or misleading, we attempted to provide a list of the most active ransomwares in history. The ransomwares may have multiple versions and different names, which makes it hard to trace them back. Table 5 is created on the basis of the publicly available information about these ransomwares.

4 Life cycle of the ransomware attack

The ransomware life cycle demonstrates the ecosystem designed and created by cyber-criminals. The "creator" and the "campaigner" of the ransomware attack work together in a highly cooperative environment and can be easily observed in the ransomware's life cycle (refer figure 5). The creator is the coder that designs and develops the malicious code of the ransomware, while the campaigner is a person or the group that organizes the attacking campaign by disseminating the malicious code.

Irrespective of different types of attacks, that is, the attack on a group or particular target, there are mainly five distinct phases in a ransomware attack [12]. Understanding the working of ransomware in each phase is helpful to identify the indicators of compromise (IOCs) and increase the probability of successful detection of a ransomware attack. The phases can be elaborated as follows.

Table 5: Evolution of ransomware samples.

Year	Ransomware samples	Infection vector	Encryption algorithms
1989	PC Cyborg (AIDS Trojan)	Floppy disk	Symmetric cryptography
2005–2010	Gpcode (2005)	Email attachment	RSA encryption
	Archiveus (2006)	Freeware downloads from the compromised websites, spam email attachments	RSA encryption
	Cryzip (2006)	Email attachment	It uses a zip library to store files in compressed format
	Gpcode.AK (2008) a variant of GPcode	Email attachment	1,024 bit RSA algorithm
2011–2015	Ransom32 Ransomware as a service (RaaS)	Distributed as a file with ".scr" extension	The data is encrypted using the AES-128 algorithm, whereas the key is encoded with the RSA technique and a public key obtained from the Command and Control Center server
	Petya (2016)	Email attachments, MEDoc, EternalBlue Exploits, Server Message Block network spreading techniques	The Master Boot Record (MBR) is overwritten by its malicious code, and Master File Table (MFT) with other user files are encrypted using the AES-128 algorithm. The key used by AES-128 is further encrypted by the RSA-2048 algorithm.

(continued)

Table 5 (continued)

Year	Ransomware samples	Infection vector	Encryption algorithms
2016–2020	Linux.Encoder.2 (2016)	A security flaw in Magento content management system (CMS)	The data is encrypted using the AES-128 algorithm, whereas the key is encoded with the RSA technique
	Crypy (2016)	A security flaw in Magento CMS	AES-256 encryption algorithm
	Locky (2016)	An email with an attached Microsoft Word document that contains a malicious macro	RSA-2048 and AES-128
	Bitpaymer (2017) (a.k.a *wp_encrypt*)	Phishing campaign, software download via a faked URL, brute force RDP attacks, and so on	RC4 and RSA-1024 encryption algorithms
	Globe3 (2017)	Corrupted email attachments that may use social engineering techniques like fake tax returns or receipts to convince computer users to open corrupted email attachments. There are other ways in which the Globe Ransomware can be delivered, including corrupted online advertisements, exploit kits, and the direct hacking of the victim's computer.	AES-256 encryption algorithm
	WannaCry (2017)	EternalBlue, doublepulsar dropper, Windows Server Message Block protocol	RSA and AES algorithms

GandCrab (2018)	Malicious advertisements and pop-ups, and email attachments	RSA, AES, and RC4 algorithm. GandCrab version 4 used Tiny Encryption Algorithm (TEA)
Ryuk (2018)	Spam email containing malicious macro as an attachment, compromised web links, or by access to a system via remote desktop services	RSA and AES encryption algorithm
REvil (2019), also known as Revil/ Sodinokibi ransomware	At first, the malware propagated via vulnerabilities in Oracle WebLogic Server. Later, the malware started spreading through phishing and spam emails, RDP servers, and scan and exploit kits	AES and Salsa20 encryption
Snake (2020)	Malicious email attachments, installing software from untrusted sources	AES-256 and RSA-2048 algorithms

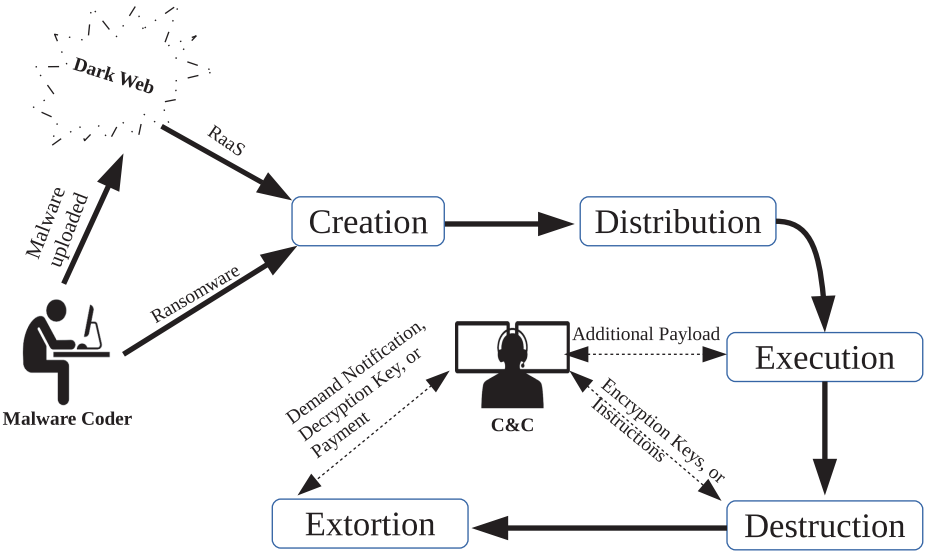


Figure 5: Life cycle of ransomware attack.

4.1 Creation

In this phase, the malware creator uses a programming language to write a new code or improve the existing code, where the selection of the programming language depends on the objective and target of the attack. These malicious codes are occasionally uploaded to the dark web so that they can be offered as a paid service to those who do not want to code the malware. "C" is a general-purpose programming language that can be used to write all sorts of malware for different computing environments such as desktop, server, and grid computing. JavaScript, VB Script, and Java are often used for computer exploits and to hit Android-based mobile devices. For remote attacks on servers, the Python language is popular among hackers. It is simpler than the C language and it does not need to be compiled into object code. For advance and complex malwares, C#, C++/VC++, and other high-level languages are generally used. The hacker need not be the coder, hence, he or she can purchase the code of malware from the dark web. These codes are generally available *as RaaS* for a payment.

4.2 Distribution

The primary objective of ransomware is to get its code executed on the victim's machine. These systems may be equipped with anti-malware tools or are updated by layered security solutions. Therefore, the entry of malicious code into such systems is the biggest challenge for cyber-criminals. To be successful in their destructive intent, a malware distribution strategy is planned and initiated by the campaigner. It could be according to an individual user or the institutional user. As most of the institutional users are professionally trained, they are less vulnerable to cyberattacks as compared to a layman, who is less computer-savvy and generally use an insecure or outdated computer. The campaigner may use a different toolkit to search for a vulnerability or security loophole in the targeted system, social engineering methods, and play with human psychology of fear to initiate the downloading of malicious code. Once the system's weakness is exploited and the user is trapped by the campaigner, the actual ransomware executable is delivered to the victim's system using a customized encrypted channel, which makes it difficult to recover and analyze over the communication medium.

4.3 Execution

Once reached its target system, another objective of ransomware is to hide its presence till its code is not executed and the persistence mechanism is not activated. It helps the malware to resume its process even after the affected system is restarted or shut down in the middle of the infection process.

To achieve persistence, they can misuse the browser extensions, exploiting dynamic link library (DLL) search order, or modify the registry keys. Some of the most common Windows registry keys, which are generally modified by the malwares including the ransomwares, can be listed as Run/RunOnce Keys, BootExecute Key, Userinit Key used by WinLogon Process, notify subkeys used to notify event handles, Explorer.exe pointed by shell key, Startup Keys, File Association keys, and so on. Furthermore if required, then the ransomware may connect to a remote Command and Control Center (C&C) to download additional payload.

4.4 Destruction

Once the ransomware is executed, it searches the backup files and removes them so that they cannot be used to restore the system. This behavior is unique to ransomware, while other malwares do not take interest to delete the backup files. It is observed on the windows system that the "vssadmin" tool is generally used by ransomwares to delete the volume shadow copies. Next, the ransomware may contact the C&C for

encryption keys. The ransomware then starts searching valuable files stored in the victim's system and encrypts them using more sophisticated encryption techniques. The encryption algorithm used may be different for different ransomware variants. The "SamSam" ransomware encrypts all files locally and does not communicate with the C&C center for the encryption key. CryptoWall v3 does not encrypt filename, CryptoWall v4 randomizes filename. The communication with a command and control server can be monitored as an IOC. However, the absence of this occurrence does not rule out the presence of malware.

4.5 Extortion

After the encryption of important files, the ransomware displays a pop-up notification to the user and asks for a ransom. The notification specifies necessary details like the amount of the ransom, the mode, and the deadline of the payment. The ransom is generally demanded in crypt-currency which makes it hard to trace. For example, the "WannaCry" ransomware demanded a ransom of $300 in Bitcoin. Depending on the ransomware family, if the deadline is missed, either the ransom is doubled or the files may be deleted permanently. Finally, the malware cleans itself off the victimized system so as not to leave behind significant forensic evidence that would help build better defenses against the malware.

5 Handling a ransomware attack

The ransomware attack can be managed by a two-step solution. First, the user must adopt preventive measures to stop the intrusion of ransomware. In addition, there should be ransomware detection and a recovery solution. These two steps can be elaborated as follows:

5.1 Preventive measures to deal with ransomware attack

There are the following four tips for individuals and businesses attempting to avoid becoming infected with ransomware, as well as how to deal with the virus if it occurs:

– *Off-line back-up of the systems*: Because ransomware destroys backup files while encrypting ordinary files, it is critical to regularly back up all important documents to a location that is not connected to the network and out of the malware's reach. This offline backup storage is to be updated and verified so

that it can be utilized as and when required. There is no need to pay a ransom to recover data if it is securely backed up in a secure location.

- *Avoid email links and attachments*: Ransomware is commonly distributed using social engineering methods, and therefore, all computer users must join the awareness program to educate themselves about phishing attacks, suspicious email attachments, and compromised advertisements, which will be useful to avoid ransomware.
- *Eliminating vulnerabilities from the system*: The outdated systems may have several loopholes that can be used by the malwares to intrude into the system. Therefore, these vulnerabilities must be removed by a regular update of the system. Patching and updating the OS, browsers, and security software should all be done regularly. Third-party plug-ins, such as Java and Flash, must also be patched if they are to be used at all.
- *Incident response plan*: Organizations should also create a ransomware incident response strategy that details the exact measures that should be taken as soon as it is discovered that a ransomware assault is underway. This will aid in ensuring a timely and effective response to avert significant damage.

5.2 Detection and recovery of ransomware attack

5.2.1 Detection

If enterprises can discover malware early enough in the case of an attack, they can reduce the harm. According to Khammas [17] and Chittooparambil et al. [18], none of the available approaches can identify or stop this form of attack, and halting such attacks is challenging. A good defense against first exploitation and infection is to update an intelligent system or other network devices with malware signatures and IOCs to block or at least alert to the existence of anomalies associated with ransomware in network traffic. These signatures and IOCs are typically malware version dependent, and they may vary over time, posing a problem for malware detection system developers.

The defense system has been developed since the beginning of malware attacks on computer systems. Traditional methodologies such as signature-, rule-, graph-, and entropy-based models are used in the majority of these methods. These systems are insufficient to detect sophisticated second-generation malware, necessitating the introduction of new techniques to solve the problem.

Malware samples are examined to extract traits that can be used to identify them. In general, there are two fundamental malware analysis techniques: static analysis and dynamic analysis; however, some detection systems employ hybrid analysis, which mixes static and dynamic analysis. These methods of analysis, as well as the tools that were utilized to conduct them, are emphasized further.

Static analysis

Static analysis of malware is conducted to extract the syntax or structural features of malware. For example, Byte n-gram, Opcode n-gram, and Portable Executables are the static features that could be extracted using static analysis. These features are used by the malware detection system to identify the malware without executing its code. Some of the tools, which can be used for static analysis, are presented in Table 6.

Table 6: Static analysis tools.

Tools	Description
PEView	It is a lightweight program to extract information from the Portable Executable (PE) header. It allows you to explore the structure and content of 32-bit PE and Component Object File Format files quickly and easily. Weblink: http://www.wjradburn.com/software/
PEid	It is a simple application that detects popular PE file packers, cryptors, and compilers. Its official website is not active but can be downloaded from other online sources.
PEStudio	It is a PE analysis tool that is developed by Marc Ochsenmeier, and is free for noncommercial use. Weblink: https://www.winitor.com/
CFF Explorer	It is a PE Editor created by Daniel Pistelli that is included in the NTCore Explorer Suite. Weblink: https://ntcore.com/?page_id=388
FileAlyzer	FileAlyzer displays simple file information, a typical hex viewer, and a variety of customized representations for extremely sophisticated file structures to help you understand what a file is for. Weblink: https://www.safer-networking.org/products/filealyzer/
IDA Pro	It is a commonly used disassembler tool. The executable file is translated to assembly code, which is then manually inspected to determine malicious behavior. Weblink: https://hex-rays.com/ida-free/

Although static analysis can quickly and effectively evaluate harmful code, it is vulnerable to code obfuscation. The simple inclusion of dead instructions can cause a mismatch with malicious codes that have already been discovered. Furthermore, when the code is encrypted, the analysis is rendered useless. No effective method exists for decrypting a cryptographic system using brute force. It is also worth noting that static analysis is not very effective against multiphase attacks. The original code could merely be a technique to open a backdoor, allowing subsequent malicious scripts to be downloaded and installed, and so may not be hazardous.

A few analysts have offered static analysis-based strategies for detecting ransomware attacks. The research on detecting malicious code based on static properties may be found in [19, 20].

Dynamic analysis

The malicious code is executed in a controlled and monitored environment, usually a sandbox, in dynamic analysis, also known as behavioral-based analysis. All actions are recorded and analyzed. API calls, system calls-based features, and information flow-based features are only a few examples. Furthermore, the dynamic analysis does not convert binary code to assembly code. The tools that can be used for dynamic analysis are briefly described in Table 7.

Table 7: Dynamic analysis tools.

Tools	Description
ProcMon	The activity of ongoing processes can be captured by this tool. It records all file system operations, registry modifications, memory operations, and network activities. Weblink: https://docs.microsoft.com/en-us/sysinternals/downloads/procmon
Wireshark	Wireshark is a network traffic analysis tool. It can collect all incoming and outgoing packets between the system and the outside world. Weblink: https://www.wireshark.org/download.html
TCPdump	It is a command-line tool for analyzing real-time network data. It is a typical packet analyzer that shows TCP/IP packets being sent and received by a computer system.
TCPview	It is a networking tool for Windows systems that displays information about all of the system's UDP and TCP endpoints. Weblink: https://docs.microsoft.com/en-us/sysinternals/downloads/tcpview
Regshot	The registry modifications produced by the executing sample are logged using the Regshot utility. The state of the Windows registry is stored both before and after the malware sample is executed. Both states are compared to see what changes the executed sample file causes. Weblink: https://regshot.en.uptodown.com/windows
Memoryze	It is a memory forensics tool that runs on the command line and can handle the entire memory dump. After launching the malware, a memory dump of the system is acquired to analyze it. Weblink: https://www.fireeye.com/services/freeware/memoryze.html
Sandboxes	A sandbox is a secure environment where users can test the suspected code without putting their device or network in danger. Many of the analysis tools listed earlier are included in the Sandboxing tool. Weblink: https://cuckoosandbox.org/

Unlike static analysis, this technique is less susceptible to code obfuscation, and the encrypted code can also be inspected. To achieve its objective and evade the detection systems, the malicious code is generally packed inside an encrypted package that must be decrypted before it can be executed, and this behavior can be identified during dynamic analysis.

Although dynamic analysis is effective against advanced and complex malware, it is both expensive and time-consuming to set up. If the study is carried out on a virtual machine, which saves cost and resources, the malware may notice and stop displaying all of its behaviors. As a result, it is critical that the environment configuration closely resembles an actual environment to effectively capture malicious behavior. Dynamic analysis is used by the majority of ransomware detection solutions [17]. Wecksten et al. [2], Takeuchi et al. [21], Vinayakumar et al. [22], and Homayoun et al. [23] employed the same methodologies to model malicious code's runtime behavior.

Hybrid analysis

Static and dynamic approaches have their own set of benefits and drawbacks. As a result, a hybrid strategy that uses both static and dynamic features in malware classification will be promising. In hybrid malware analysis, both static and runtime behaviors are captured before, during, and after the malware is executed. The authors also used a hybrid approach in [24].

Machine learning (ML)-based approach to detect ransomware

As we all know, malware development moves at a breakneck speed, and the complexity of malware writing grows by the day. As a result, traditional approaches such as rule-based, graph-based, and entropy-based cannot detect malware [11]. ML techniques are now widely used to detect new or previously unknown malware [25, 26]. These techniques can detect not only known malware but also unknown malware by using information from previously detected malware. It is a two-step process in which features like API Calls, N-grams, Strings, Opcodes, and Control Flow Graph are extracted from known datasets and play a critical role in not only representing the malicious but also speeding up the learning and classification/detection processes. In the second step, ML techniques such as "decision tree (DT)," "naive Bayes (NB)," "hidden Markov modes," and "neural networks" are trained for malware detection and classification. Figure 6 shows and exemplifies the conventional diagram of the malware detection system using ML algorithms. The benign and malicious files are used to extract the features that can be used to identify them. Then, in the next step, most relevant features that best represent them are selected. Finally, the selected features are used to train ML classification algorithms. After sufficient training, the classifier is used for the classification of the input file as benign or malicious.

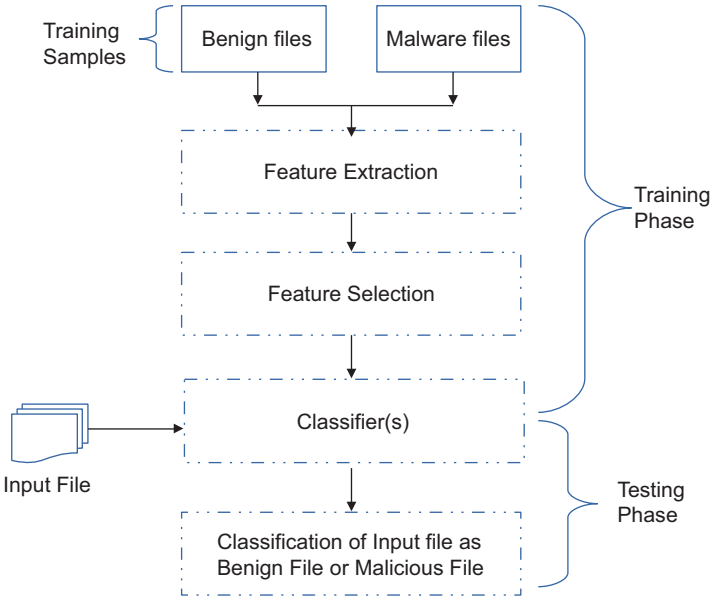


Figure 6: Flow diagram of machine learning-based malware detection system.

ML has many techniques that may be used to create a model over a set of malware attributes, which is one of the main advantages of employing it in malware detection. It can also cut down on human effort and time spent analyzing the malware.

ML, despite its advantages, is not without its drawbacks. The training and updating of malware classifiers is the first barrier in implementing ML. To detect new and mutated malware, malware detectors must be updated regularly. As a result, it is more expensive and sophisticated than others. Another issue is adversarial ML, which is a major difficulty in malware classifiers that are machine-based. When a malware creator uses ML strategies to get around a malware detector, this is referred to as adversarial ML [25].

We can minimize the dimensionality of the dataset to reduce the training cost because ML methods take longer if the dataset has more data characteristics. Feature selection and dimensionality reduction methods can be used to choose only the most valuable and discriminative virus features for this purpose.

The hybrid malware classifiers can be used to overcome the difficulty of adversarial ML. Both static and dynamic features, as well as numerous classification techniques, can be used in these classifiers. Malware detectors based on a single ML algorithm can be overcome, whereas ensemble malware detectors can handle adversarial ML better [25].

Various ML algorithms, such as "support vector machine," "NB," "random forest (RF)," "DT," "artificial neural network," "k-nearest neighbors," "logistic regression,"

and ensemble algorithms, such as "RF," "adaptive boosting," are widely used by various researchers in the literature. The recent developments in the ML-based approaches for malware detection can be found in [25, 27–29].

5.2.2 Containment

Once a system has been infected with ransomware, several steps must be taken to limit it locally so that network files are not affected. The best way to keep the infection contained is to use an endpoint protection solution that can detect and terminate the process. If ransomware is discovered, network connectivity can be stopped, preventing it from encrypting files on the network even if it manages to get to the endpoint.

5.2.3 Eradication

Following the detection and containment of the ransomware attack, the next step is to entirely remove it from the compromised systems. According to industry standards, the machines should be replaced rather than cleaned. It is impossible to know if residual files are concealed on the system and can reinfect machines, just as it is hard to know if any other sort of malware is present. Clearing network locations like mailboxes or file shares, removing infected email attachments from the mailbox, or removing ransomware instructions from the file share are sometimes more important. If the organizations chose to clean rather than replace, it is critical that they keep an eye out for signatures and other IOCs to keep the attack from resurfacing.

5.2.4 Recovery

The first step in the recovery process will be to restore from a backup. Any ransomware attack can be turned into a nonissue by simply replacing or cleaning systems and recovering from backups if there are good verified backups. There may be some downtime, but there should not be an issue that lasts more than a day. In the majority of ransomware incidents, determining what exact attack vector was used against the system is a critical step. Knowing how ransomware entered your system can help organizations better prepare their protection systems and direct their detection methods in the future.

6 Case study *of WannaCry ransomware*

WannaCry is ransomware (a.k.a *Wcry or WannaCrypt*) that spreads quickly through a variety of computer networks in May 2017, infecting over 230,000 machines worldwide and causing billions of dollars in damages. Unlike earlier ransomware attacks that spread through social engineering (unsolicited email or visiting a hacked website), WannaCry exploits worm-like qualities to propagate without the need for human intervention. It makes copies of itself using the host system's resources, then infects the rest of the network and the devices it comes in touch with. Following the infection, it encrypts all of the files and the folders on the system, making them inaccessible to the user, and then demands a ransom payment of $300–$600 in bitcoins to decrypt them. As a result, WannaCry became dangerously widespread, exponentially increasing the rate of infection. This attack had a significant impact on many companies, particularly healthcare organizations.

In this case study, we presented a static and dynamic analysis of WannaCry to determine its resources and functionalities, as well as its use of DLLs and communication protocols. A sample of the ransomware can be obtained from VirusShare [30] and must be scrutinized with great care in a virtual environment. The analysis presented in this case study is derived from the examination of WannaCry's behavior from [31–33], and serves as a foundation for the creation of sustainable ransomware defense systems.

6.1 Static and dynamic analysis of WannaCry

WannaCry components: There are two main components such as *the worm component (MD5 hash: db349b97c37d22f5ea1d1841e3c89eb4; type: PE32 executable)* and *the encryption component (MD5 hash:84c82835a5d21bbcf75a61706d8ab549; type: PE32 executable).* The ransomware also contains a number of compressed XIA resource files, which are listed in Table 8.

Table 8: WannaCry XIA resource files.

File name	Description	MD5 hash
"msg" Directory	It contains "*Readme*" RTF files in different languages with the extension "*m_*.wnry*"	m_english.wnry file: fe68c2dc0d2419b38 f44d83f2fcf232e
b.wnry	It is an image file that contains instructions for decryption as well as a ransom demand	c17170262312f3be7027bc2ca825 bf0c

Table 8 (continued)

File name	Description	MD5 hash
c.wnry	It contains TOR addresses	ae08f79a0d800b82fcbe1b43cdbd befc
r.wnry	It includes additional decryption instructions that the decrypter tool can utilize	3e0020fc529b1c2a061016dd2469 ba96
s.wnry	It is a compressed zip file that includes the executable for "Tor software"	ad4c9de7c8c40813f200ba1c2fa 33083
t.wnry	It contains encrypted ransomware DLL file	5dcaac857e695a65f5c3ef1441a73a8f
u.wnry	It contains "@WanaDecryptor@.exe" decrypter file	7bf2b57f2a205768755c07f238 fb32cc
f.wnry	A list of randomly chosen files, used for demonstration of decryption process to the victim user	–
taskdl.exe	File deletion tool	4fef5e34143e646dbf9907c4374276f5
taskse.exe	It sets out RDP sessions and executes the instance of malware on each session	8495400f199ac77853c53b5a3f 278f3e
	Additional files	
00000000.res	It contains TOR/C&C information	168d54591c029609 959eb4256cbcea26
00000000.pky	It contains generated public key	6f4e6640a2bc54a0 778130f7a25cb1b1
00000000.eky	It contains generated private key	6317124f38c33cce 36291ec3bc835db4

6.2 Life cycle of WannaCry

6.2.1 Distribution (Initial infection and propagation of WannaCry)

The vulnerability WannaCry exploits to lie in the Windows implementation of the Server Message Block (SMB) protocol. The SMB protocol helps various nodes on a network communicate, and Microsoft's implementation could be tricked by specially crafted packets into executing arbitrary code. Microsoft itself had discovered the vulnerability a month prior and had released a patch SMB protocol, but many systems remained vulnerable. WannaCry used an exploit, known as *EternalBlue*, to scans for vulnerable systems and then install the DoublePulsar dropper, *a software program that extracts embedded application components* and executes a copy of

itself. The payload is instantly transferred to the targeted system as soon as the "Doublepulsar" setup is configured. It includes "kernel and userland shellcode," as well as the "launcher.dll," which includes the mssecsvc binary in the resource section. The "launcher.dll" file is then encased in the "lsass.exe" system process and is used as the loader for "mssecsvc.exe." It solely runs in memory and does not leave any file traces on the disk. On the other hand, the "mssecsvc" services running on the infected system will try a connection with the SMB protocol of the next targeted machine. If the connection is established, then it will look for the presence of the "Doublepulsar" backdoor. If the backdoor is present, then the payload is directly uploaded to the victim's system; otherwise, the EternalBlue attack is initiated to search the vulnerable system. Figure 7 illustrates the flow of the infection process, including the initial stage of an attack initiated by the attacker as well as the infection launched through the compromised machine. All phases of WannaCry infection are illustrated in Figure 8.

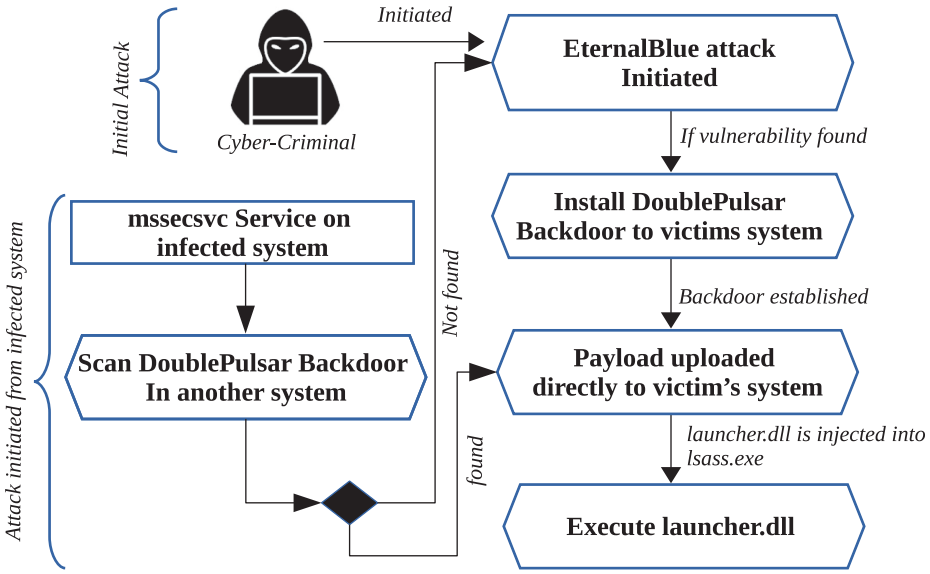


Figure 7: Initial infection.

Execution: Before the start of any operation, the ransomware tries to connect with a hard-coded kill-switch URL, "www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com." The successful connection with this domain name will cause the ransomware to stop infecting other machines. Otherwise, it will proceed to the next phase by dropping of "tasksche.exe" to the current system and keep propagating to other machines that are connected to this system. The kill-switch was designed intentionally by the attacker so

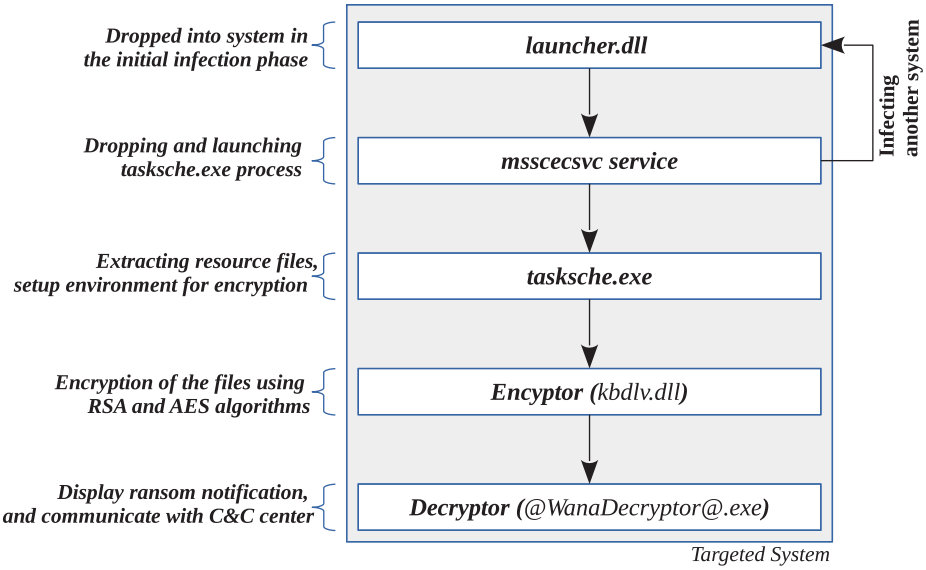


Figure 8: All phases of WannaCry infection.

that they can stop or control the infection chain of the ransomware. The URL was not registered at the launch of the attack, which causes the WannaCry to spread quickly.

The core executable file, "tasksche.exe," which is dropped by "mssecsvc.exe" services is responsible for extraction and loading of compressed resource files to a folder created for it. It is also responsible for the setup of the environment for the encryption of files and directories stored in the system. The unzipped resource files are extracted using the password "WNcry@2ol7." The extracted XIA resource files are listed in Table 6. Other responsibilities carried out by "tasksche.exe" include:

- Establish memory persistence by creation of autorun registry key.
- Take full access on files and directories.
- Write bitcoin address to "c.wnry" file.
- Decrypt "t.wnry" file to generate encryption Dynamic Link Library (dll) file in the memory and export TaskStart system thread to start the encryption process.

Destruction: To fulfill its malicious intent, the ransomwares encrypt the important files in the disk and remove or damage all possible sources of the backup. The effectiveness of any encryption process depends on the way the encryption keys are managed. The WannaCry uses Windows cryptographic libraries to employ a combination of the RSA and AES encryption algorithms. The encryption process begins with a pair of root public keys and a root private key (*known to the attacker only*). A pair of RSA-2048 public and private keys is generated and stored in *00000000.pkv* and *00000000.ekv* files, respectively. The private key is stored after being encrypted by the root public key.

The targeted user files are encrypted using AES-128 encryption keys generated at random for each file. The public key read from the "00000000.pky" resource file is also used to encode the unique AES-128 encryption key. WannaCry uses the "vssadmin" command to attempt to remove any Windows shadow copies after the encryption procedure is completed. The intermediate temporary files are periodically removed by the *taskdl.exe* process. Finally, the ransomware starts infecting other vulnerable systems and moves itself to the *extortion phase*.

Extortion: The WannaCry encrypter starts the embedded decrypter binary "@WanaDecryptor@.exe," which displays a notification, two timers, and instructions for transmitting the ransom in the infected system's configured language. It asks for $300 in bitcoins to be sent to a certain address. Although only the first was observed to be used by the tested sample, the following addresses are hardcoded in the binary:

"12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw";
"115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn";
"13AM4VW2dhxYgXeQepoHkHSQuy6NgaEb94"

If the ransom is not paid before the first timer ends, the fee doubles and the files become unrecoverable once the second timer expires. WannaCry establishes a secure HTTPS link with the Onion server (*"gx7ekbenv2riucmf.onion"*; *"57g7spgrzlojinas.onion"*; *"xxlvbrloxvriy2c5.onion"*; *"76jdd2ir2embyv47.onion"*; or *"cwwnhwhlz52maqm7.onion"*) over the Tor network to register the system and transfer encryption keys, and uses common Tor ports 9001 and 9050 for network traffic and directory information.

6.3 Indicator of compromise (IoC)

In this section, we provide a list of IoCs, which can be used to extract the features for the training of ML-based detection systems.

(a) *Execution of the command*: *"attrib +h . followed by icacls . /grant Everyone:F /T / C /Q"*
(b) *Execution of the command*: *"Service* %current_directory%5bef35496fcbdbe841c82f4d1ab8b7c2.exe -m security" (services auto start) command "C:\WINDOWS\tasksche.exe /i" *command "cmd.exe /c vssadmin delete shadows /all /quiet & wmic shadowcopy delete & bcdedit /set {default} bootstatuspolicy ignoreallfailures & bcdedit /set {default} recoveryenabled no & wbadmin delete catalog –quiet"*
(c) *Access to the URL*: *"www[.]iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com"*
(d) *Registration of registry key*: *"HKLM\SOFTWARE\Wow6432Node\WanaCrypt0r\wd"*

7 Conclusion

Everywhere, computers or embedded digital devices are employed to do tasks faster and more correctly without the need for human intervention. Because of its broad breadth, attackers are more likely to compromise computer systems, making cybersecurity a crucial component of cyberspace. Malwares are now widely recognized as being extremely complicated and are evolving at a rapid rate. These malicious codes are used not just to annoy individuals, steal and erase important information, but also by organizations and cyber-criminals to achieve their objectives. Ransomware threats on enterprises are just getting started. Because the offenders profit so much from these attacks, they will undoubtedly become more common, more harmful, and more costly. An organization's ability to defend against a ransomware campaign depends on its level of preparedness and its ability to detect, terminate, and limit suspicious activities.

This chapter discusses malware's evolution, current state, and analysis methodologies. It also disseminates information on various obfuscation methods employed by malicious software, as well as an ML-based approach to combat them. It also includes a case study of the WannaCry ransomware, which may be beneficial in developing effective and efficient mitigation techniques for WannaCry and other ransomware with comparable characteristics.

References

[1] Butler, A. M. (2007). William Gibson: Neuromancer. In *A companion to science fiction*.

[2] Wecksten, M., Frick, J., Sjostrom, A., & Jarpe, E. (2017). A novel method for recovery from Crypto Ransomware infections. In 2016 2nd IEEE International Conference on Computer and Communications, ICCC 2016 – Proceedings (pp. 1354–1358).

[3] Qucik Heal Annual Threat Report 2019. www.quickheal.com" [Online]. Available: www.quick heal.com

[4] McAfee. (2019). McAfee labs threats report: August 2019. In *Computer fraud & Security*.

[5] Savage, K., Coogan, P., & Lau, H. (2015). Symantec SECURITY RESPONSE The evolution of ransomware. (p. 57), [Online]. Available: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the-evolution-of-ransomware.pdf

[6] Avira. Malware Threat Report: Q3 2020 Statistics and Trends. [Online]. Available: https://www.avira.com/en/blog/malware-threat-report-q3-2020-statistics-and-trends

[7] Bitdefender. (2020). Mid-Year Threat Landscape Report. [Online]. Available: https://www.bitdefender.com/files/News/CaseStudies/study/366/Bitdefender-Mid-Year-Threat-Landscape-Report-2020.pdf

[8] Sibi Chakkaravarthy, S., Sangeetha, D., & Vaidehi, V. (2019). A survey on malware analysis and mitigation techniques. *Computer Science Review*, *32*, 1–23. [Online]. Available:. https://linkinghub.elsevier.com/retrieve/pii/S1574013718301114

[9] Aslan, O., & Samet, R. (2020). A Comprehensive Review on Malware Detection Approaches. *IEEE Access*, *8*, 6249–6271. [Online]. Available:. https://ieeexplore.ieee.org/document/8949524/

[10] Qamar, A., Karim, A., & Chang, V. (2019). Mobile malware attacks: Review, taxonomy & future directions. *Future Generation Computer Systems*, *97*, 887–909. [Online]. Available:. https://linkinghub.elsevier.com/retrieve/pii/S0167739X18331601

[11] Sahay, S. K., Sharma, A., & Rathore, H. (2020). Evolution of malware and its detection techniques. *Information and Communication Technology for Sustainable Development*, 139–150. [Online]. Available:. http://link.springer.com/10.1007/978-981-13-7166-0_14

[12] Kok, S. H., Abdullah, A., Jhanjhi, N. Z., & Supramaniam, M. (2019). Ransomware, threat and detection techniques: A review. *IJCSNS International Journal of Computer Science and Network Security*, *19*.

[13] Bhardwaj, A. (2016). Ransomware: A rising threat of new age digital extortion. In *Online banking security measures and data protection* (pp. 189–221).

[14] Statcounter Global Stats. (2021). [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide

[15] Yang, W., Kong, D., Xie, T., & Gunter, C. A. (2017). Malware Detection in Adversarial Settings. In Proceedings of the 33rd Annual Computer Security Applications Conference (pp. 288–302) [Online]. Available: https://dl.acm.org/doi/10.1145/3134600.3134642

[16] Statista. [Online]. Available: https://www.statista.com

[17] Khammas, B. M. (2020). Ransomware detection using random forest technique. *ICT Express*, *6*(4), 325–331. [Online]. Available:. https://linkinghub.elsevier.com/retrieve/pii/S2405959520304756

[18] Chittooparambil, H. J., Shanmugam, B., Azam, S., Kannoorpatti, K., Jonkman, M., & Samy, G. N. (2019)A Review of Ransomware Families and Detection Methods. In 3rd International Conference of Reliable Information and Communication Technology (IRICT 2018), Kuala Lumpur, Malaysia (pp. 588–597). [Online]. Available: http://link.springer.com/10.1007/978-3-319-99007-1_55

[19] Zhang, H., Xiao, X., Mercaldo, F., Ni, S., Martinelli, F., & Sangaiah, A. K. (2019). Classification of ransomware families with machine learning based on N-gram of opcodes. *Future Generation Computer Systems*, *90*, 211–221. [Online]. Available:. https://linkinghub.elsevier.com/retrieve/pii/S0167739X18307325

[20] Baldwin, J., & Dehghantanha, A. (2018). Leveraging Support Vector Machine for Opcode Density Based Detection of Crypto-Ransomware. In Cyber Threat Intelligence (pp. 107–136). [Online]. Available: http://link.springer.com/10.1007/978-3-319-73951-9_6

[21] Takeuchi, Y., Sakai, K., & Fukumoto, S. (2018). Detecting Ransomware using Support Vector Machines. In Proceedings of the 47th International Conference on Parallel Processing Companion (pp. 1–6). [Online]. Available: https://dl.acm.org/doi/10.1145/3229710.3229726

[22] Vinayakumar, R., Soman, K. P., Senthil Velan, K. K., & Ganorkar, S. (2017). Evaluating shallow and deep networks for ransomware detection and classification. In 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 259–265). [Online]. Available: https://ieeexplore.ieee.org/document/8125850/

[23] Homayoun, S., Dehghantanha, A., Ahmadzadeh, M., Hashemi, S., & Khayami, R. (2020). Know abnormal, find evil: Frequent pattern mining for ransomware threat hunting and intelligence. *IEEE Transactions on Emerging Topics in Computing*, *8*(2), 341–351. [Online]. Available:. https://ieeexplore.ieee.org/document/8051108/

[24] Subedi, K. P., Budhathoki, D. R., & Dasgupta, D. (2018). Forensic Analysis of Ransomware Families Using Static and Dynamic Analysis. In 2018 IEEE Security and Privacy Workshops (SPW) (pp. 180–185). [Online]. Available: https://ieeexplore.ieee.org/document/8424649/

[25] Singh, J., & Singh, J. (2021). A survey on machine learning-based malware detection in executable files. *Journal of Systems Architecture*, *112*, 101861. [Online]. Available:. https://linkinghub.elsevier.com/retrieve/pii/S1383762120301442

[26] Ndibanje, B., Kim, K., Kang, Y., Kim, H., Kim, T., & Lee, H. (2019). Cross-method-based analysis and classification of malicious behavior by API calls extraction. *Applied Sciences*, *9*(2), 239. [Online]. Available:. http://www.mdpi.com/2076-3417/9/2/239

[27] Gibert, D., Mateu, C., & Planes, J. (2020). The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *Journal of Network and Computer Applications*, *153*, 102526. [Online]. Available:. https://linkinghub.elsevier.com/retrieve/pii/S1084804519303868

[28] Abijah Roseline, S., & Geetha, S. (2021). A comprehensive survey of tools and techniques mitigating computer and mobile malware attacks. *Computers & Electrical Engineering*, *92*, 107143. [Online]. Available:. https://linkinghub.elsevier.com/retrieve/pii/S0045790621001464

[29] Ucci, D., Aniello, L., & Baldoni, R. (2019). Survey of machine learning techniques for malware analysis. *Computers & Security*, *81*, 123–147. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167404818303808

[30] ViRus Share malware repository. [Online]. Available: https://virusshare.com

[31] Chen, Q., & Bridges, R. A. (2017). Automated Behavioral Analysis of Malware: A Case Study of WannaCry Ransomware. In 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 454–460). [Online]. Available: http://ieeexplore.ieee.org/document/8260673/

[32] Akbanov, M., Vassilakis, V. G., & Logothetis, M. D. (2019). Wanna Cry ransomware: Analysis of infection, persistence, recovery prevention and propagation mechanisms. *Journal of Telecommunications and Information Technology*, *1*, 113–124. [Online]. Available: https://www.il-pib.pl/czasopisma/JTIT/2019/1/113.pdf

[33] Satheesh Kumar, M., Ben-Othman, J., & Srinivasagan, K. G. (2018). An Investigation on Wannacry Ransomware and its Detection. In 2018 IEEE Symposium on Computers and Communications (ISCC) (pp. 1–6). [Online]. Available: https://ieeexplore.ieee.org/document/8538354/

Samuel Wedaj Kibret*

Property-based attestation in device swarms: a machine learning approach

Abstract: Cyber-physical systems (CPS) and industrial Internet of things (IIoTs) are characterized by large numbers of tightly integrated heterogeneous components in a network. As such devices become imperative in mission-critical systems, their security is of immense concern. Toward this end, remote attestation (RA) schemes are widely used to verify the integrity of devices in IIoTs/CPS. However, such interactive mechanisms are vulnerable from verifier impersonation to replay and denial-of-service attacks. In addition to that, (binary) RA techniques are not resilient as any software upgrades would result in platform configuration and hash value changes. This necessitates repeated exchange of genuine hash values between collaborating devices. The problem is exacerbated because device swarms work in a group to attain a common goal, which is better identified by the system's overall behavior.

Training member devices using machine learning models and attesting them via property-based techniques help to increase the attained security and privacy performance of an Internet of things (IoT) swarm. In this chapter, to enhance the security guarantees of IoT networks and ensure swarm resiliency, we provide a comprehensive survey on the properties of IoT swarms. We also provide a concise overview of issues of IoT network attestation that arise from the fundamental nature of the underlying system, namely, the number of devices to be attested, the nature of member devices, and the property to be verified. We also describe a full prototype implementation and evaluate performance using OP-TEE, an Advanced RISC machine TrustZone-based open-source implementation of trusted execution environment. We also assess performance and analyze security for large swarms and show that the proposed approach is very effective and robust against various attacks.

Keywords: Property-based attestation, cyber-physical system, machine learning, Internet of things

1 Introduction

An embedded device is an object that is designed for a specialized computing task. It is small in size and is built for specific tasks. It is designed to achieve maximum

***Corresponding author: Samuel Wedaj Kibret,** Indian Institute of Technology Delhi, India, email: samuel.wed@cse.iitd.ac.in, samuel_wed@yahoo.com

https://doi.org/10.1515/9783110766745-004

effectiveness and efficiency to fulfill various constraints such as timing, power, performance, and cost. It is generally characterized by its resource-constrained nature in terms of hardware resources such as central processing unit power, RAM memory, and graphics processing unit power. These devices are usually embedded into other objects in order to provide a greater purpose. In the ever-increasing connected world of computing systems, such devices interconnect over the Internet and form a network of interconnected objects, often referred to as the Internet of things (IoT).

Embedded devices are used in the ever-growing network of cyber-physical systems (CPS) and industrial IoT (IIoT). These networked systems are applied in the military, industries, business, medical field, retail, transportation, in-home automation, and other safety and security-critical applications. They generate, process, and exchange a vast amount of mission-critical and privacy-sensitive information, thus becoming targets of various attacks.

Devices in IoTs and CPS can operate in swarms and are connected to each other [1]. They are large clusters of devices working together to perform some task. Their key characteristics are that, they are smart, mobile, and interconnected and operate in large numbers. Application scenarios with networks of faulty links and/or swarms of dynamic topologies such as target recognition using unmanned aerial vehicles [2, 3], robotic systems (robot swarms) [1], and self-driving connected cars (vehicular ad hoc networks) [4–6] exhibit temporary device disconnections, and topology changes with highly dynamic, unstructured mobile networks.

Recently, a number of high-profile threats and attacks targeting IoTs have been reported in the literature. Attackers pose a significant amount of threats to IoTs either by injecting malware [7–10] or through observing properties (i.e., execution time, memory usage patterns, task schedule, power consumption, etc.) of the software running [11] on the underlying device. Such attacks could affect the integrity of the software running on the device as well as the underlying system's firmware or can be used to reveal the running application's secrets. Devices under attack could stop working, behave differently, or be leveraged to pose distributed denial-of-service (DoS) attacks.

In addition to that, recent studies [12, 13] revealed that embedded software suffers from a variety of vulnerabilities that can be exploited at runtime. These vulnerabilities are in the form of buffer overflow, format string vulnerability, integer overflow, double free, and so on. Such memory corruption attacks could overwrite a function pointer or its return address to generate unintended activity (usually harmful) without injecting any malicious code [14]. These types of attacks are called return-oriented programming attacks.

1.1 Remote attestation of embedded devices

Remote attestation (RA) in embedded devices is a process where a target system (i.e., prover) makes a claim about its internal state by supplying reliable evidence to a

verifier (trusted party). The verifier will then decide whether the target system's attestation information is within its acceptable (secure/trustworthy) state (see Figure 1).

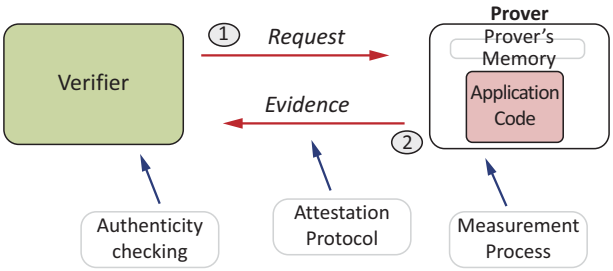


Figure 1: Remote attestation.

Based on their technology, current attestation techniques can be divided into three groups: hardware-based, software-based, and hybrid. In light of protecting the prover's report from being forged by an adversary, some techniques use secure hardware [15, 16], while others are based on trusted software [17, 18]. Secure hardware is too costly and complex for low-end embedded devices. On the other hand, trusted software relies on strong adversarial and algorithm optimality assumptions, which are very difficult to achieve in reality [19].

The hybrid (also called software/hardware co-design) attestation techniques use minimal hardware support as mentioned in references [20, 21]. Defrawy et al., in SMART [20], present secure attestation with minimal hardware changes in memory controller unit (MCU). In SMART, each device embeds an immutable code and key realized by storing them in read-only memory. SMART, additionally utilizes MCU to provide access control to the memory region to only the secure code, thus guaranteeing its contents' privacy and security. Trustlite [21], a follow-up work on SMART, generalizes this idea to introduce execution-aware memory access control. TyTAN [22] extends TrustLite's approach with a dynamic configuration of access control rules, albeit it incurs additional architectural complexity. However, they require additional hardware to ensure the atomicity of the execution of attestation code as well as to store attestation-related keys.

On the other hand, trusted execution environment (TEE) is a hardware-assisted security solution that provides an isolated environment in which protected application executions are carried out. TEE ensures security by allowing data to be stored and executed in a trusted environment. Nowadays, various IoT devices are equipped with TEE solutions [23]. Among the solutions, the most common ones are Advanced RISC machine (ARM) TrustZone [24] and Intel Software Guard Extension [25].

1.2 Outline

The rest of the chapter is organized as follows. In Section 2, we detail the problem and describe the attack model. Section 3 discusses our attestation protocol in detail, while Section 4 presents its implementation and performance evaluation. In Section 5, we present security considerations of our approach, and in Section 6, we describe and compare our work with other state-of-the-art schemes. We discuss protocol extensions in Section 7 and finally conclude with a discussion in Section 8.

2 Problem definition, attack model, and assumptions

Attacks targeting (software) on embedded devices have become pervasive [26]. These attacks could modify the underlying program and/or alter the runtime behavior of the running code without changing its binary. Adversary, ***Adv***, through the former, tampers with the software to manipulate its inputs.

We consider software-only attacks (remote and local adversaries), that is, adversary, ***Adv***, cannot physically tamper with devices in the swarm.

In addition to these, classic swarm attestation techniques rely on one trusted entity called the operator, ***O***, that manages all attestation-related activities. Specifically, swarm members interact with the ***O*** to submit their attestation reports and/or inform what is happening in the devices they interact with (i.e., if they have been compromised) However, as shown in [27–29], having an ***O*** creates a single point of failure for security. In addition to that, in a swarm of self-organizing IoT networks, each member device independently changes its position (i.e., it enters and leaves the network continuously). Thus, it becomes very challenging for the ***O*** to track these mobile devices.

The proposed approach requires a secure and isolated execution environment as provided by the ARM TrustZone [24]. We target an IoT swarm with ARM-based IoT devices. We also assume that both verifier and prover have access to the attested embedded program **P**.

We also assume that devices are initialized and deployed by a trusted device called Operator ***O***. Note that devices in IoTs and CPS operate in swarms and are connected to each other.

3 The proposed approach: ML for property-based attestation

RA relies on providing a status report of one device to a trusted party to reveal that it is working as per the specification. Note that the proposed approach verifies the runtime behaviors of the target program, **P**, through checking execution time properties such as the number of *Instructions*, execution *cycles* it takes, and the number of *branch* instructions observed. Our protocol essentially has two phases to maintain these features: (execution time) property fingerprinting and attestation. We first collect execution time properties (i.e., the number of *instructions*, number of *cycles*, and *branch* instructions) by running the target program **P** for program input *i* from the verifier. After that, a machine learning (ML)-based attestation is conducted to check if P in prover is safe from runtime attacks.

3.1 Property fingerprinting

The prover runs **P** using *i* as an input in the first phase. Here, the prover produces valid inputs to the measurement entity (i.e., runtime information) by tracing each instruction of the target program. We use the same instrumentation approach as in C-FLAT [30] to collect execution time properties (also called behaviors). We track properties by collecting every instruction into our runtime reference table data structure. That is, we count the number of instructions and collect and track the number of instruction cycles overserved through monitoring memory and the program counter's (PC) values.

In the case of branch instructions, we keep program execution flow-related information by storing the source and destination addresses of the underlying branch instruction. Remember that we use ARM-based devices. Thus, as per ARM Architecture Procedure Call Standard [31], branch instructions (subroutines) are called using **bl** (Branch with Link) or **blx** (Branch with Link and eXchange) instructions. They call a branch instruction by putting its address on the PC and the return address to the **lr** (link register). Inside a subroutine, it uses **b** (Branch) and **bx** (branch and eXchange) instructions to write into PC. However, it retains the **lr** value untouched (for the original subroutine call) and uses instructions that write to PC as return instructions.

Algorithm 1 illustrates the pseudocode of an algorithm that traces the runtime properties. Note that control flow-inducing instructions include direct/indirect branches, and conditional and loop instructions.

Algorithm 1: Runt-time property Fingerprinting

Input: Target Program (*P*)
Output: instTotal (Total Instructions), cyclesTotal (Total Cycles), Control Flow Set (*CF*)

```
instTotal ← cyclesTotal ← CF ← RT ← ∅
for each instruction in P do
    instTotal = instTotal + 1
    if inst is storeInstruction or loadInstruction then
        /*instruction has finished execute phase*/
        cyclesTotal = cyclesTotal + 1
    end
    if inst is loadInstruction then
        CD ← {{inst Id_1, ...}, memPos} ∈ RF
        /* instID_1 is storing instruciton st_1 */
    end
    if inst is conditionalInstruction then
        CF ← CF ∪ {Src, Dst}
    end
    if inst is loopInstruction then
        L_in ← L_in + 1
        /*Loop count for instruction in is incremented */
        CF ← CF ∪ {Src, Dst}
    end
end
```

3.2 ML for property-based attestation

Verifier initiates attestation by sending program input i (i.e., an input to runtime attestation), Nonce N to prevent replay attack, and attestation session identifier s. The attestation program on the Prover executes the target program, does property fingerprinting, provides the attained profile to the ML classifier, and generates an attestation report.

We first collect the necessary data for training our ML classifiers by executing sample MiBench embedded benchmark suite [32, 33] IoT applications, that is, we generate and capture the profile of the normal execution of the underlying sample applications. The MiBench embedded benchmark suite comprises six different representative embedded categories: Automotive and Industrial Control, Network, Security, Consumer Devices, Office Automation, and Telecommunications. They exhibit a much wider variance in behavior as well as size.

3.2.1 ML Classifier

We use a KNN (k-nearest neighbors, supervised) learning model [34], where training data are used when making predictions to classify the data. KNN is one of the simplest ML models that makes it the best fit for attestation in IoT devices.

Further, this model effectively detects attacks with a very minimum false-positive rate [35]. In addition to that, its response (prediction) time is very fast with a high

level of accuracy [36], making it more suitable for device attestation in IoT swarms. Note that we use 5 as the K value since it helps us make predictions accurately.

Figure 2 illustrates our attestation protocol and the corresponding message exchanges between Verifier and Prover.

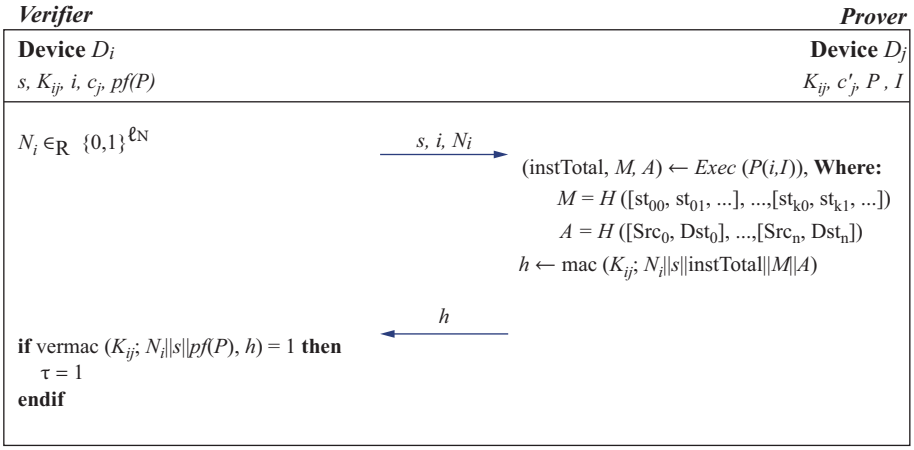


Figure 2: Attestation protocol.

Verifier initiates attestation by sending program input **i** (i.e., an input to runtime property-based attestation), Nonce **N** to prevent replay attack, and attestation session identifier **s**. The attestation program on the Prover executes the target program, collects runtime data (property fingerprints), and generates an attestation report. The attestation report essentially comprises the total number of instructions (*instTotal*), cycles (i.e., **M** in Figure 2), and branches (i.e., ***A*** in Figure 2). Upon the arrival of the prover's report, the verifier extracts the report and decides whether the target system's (i.e., the prover's) attestation information indicates that it is in its acceptable state. For this, the verifier executes KNN with its own **pf** (expected property fingerprint for input i), and compare it with the hashed value **h** comprising the prover's *instTotal*, **M** and ***A***.

Table 3 (in notation A) provides an overview of the variables and parameters used in the description and evaluation of the proposed protocol.

3.3 Decentralized swarm attestation

Devices in IoTs and CPS operate in swarms and are connected to each other. A trusted external entity/operator, ***O***, can initiate swarm attestation by contacting a single device in the swarm.

Consider a connected network (i.e., swarm) of **z** low-end devices, where ***O*** has direct contact with device 1, D_1. Suppose that D_1 is a root device from which all other swarm members descend. First, ***O*** sends attestation request to D_1, and verifies the replied attestation report. After successful verification, D_1 sends attestation request to its neighbors and checks if there is runtime and/or static attack on the target program in them or not. This way a trusted device (the verified one) checks each of its neighbors that are yet to be attested. This means that the previously verified child devices can run the attestation protocol in parallel until all the z devices get verified. If a compromised device is found, then the verifier (device appointed for that particular attestation) broadcasts an error message so that other devices are aware of its presence. This way, malicious devices leave the group, and genuine devices remain in the swarm. Subsequent attestations can be instantiated by any genuine member as discussed in [35]. It also assures system resilience after device compromise or presence of faulty devices in the swarm.

Note that the swarm is formed by establishing common attestation keys between neighboring devices, as described in [37]. This key is unique for each pair of devices. For example, attestation key K_{ij} is unique for devices D_i and D_j (see Figure 2 above).

4 Implementation and performance evaluation

ARM TrustZone is a (hardware) security extension available in devices with Cortex-based processor systems [38]. ARM TrustZone partitions processing environments into two as *Normal* and *Secure Worlds*. These two environments have their own address spaces and different privileges. The *Secure World* can access the code and data in the Normal World while its processing is immune from anything in the *Normal World*.

4.1 Proof of concept: ARM-based implementation

We implement our protocol on OP-TEE [39, 40], which is an ARM TrustZone-based open-source implementation of TEE . The target program's software stack P runs in the *Normal World* while attestations are carried out in TrustZone's *Secure World*.

Figure 3 shows our prototype implementation (on ARM), where the target program's execution and execution time property-related data collection are in the *Normal World*, while attestation is taking place in the *Secure World*. During program execution, the runtime tracer extracts each instruction, the cycles, and the branches as described in Section 3.1. On the other hand, the measurement and hash engines are located in the *Secure World* so that the trust anchor protects the execution of the attestation protocol.

4.2 Performance evaluation

We use MAC-based BLAKE2 (i.e., a keyed cryptographic hash function BLAKE2) [41, 42] due to its speed, simplicity, and security. As attestation-related operations are carried out in TEE, we evaluated performance overhead by measuring the time spent in ARM TrustZone's *Secure World*. Each operation results in the execution of the underlying instructions in a trusted (secure) mode. This means that control swayed back and forth between Normal World and *Secure World*.

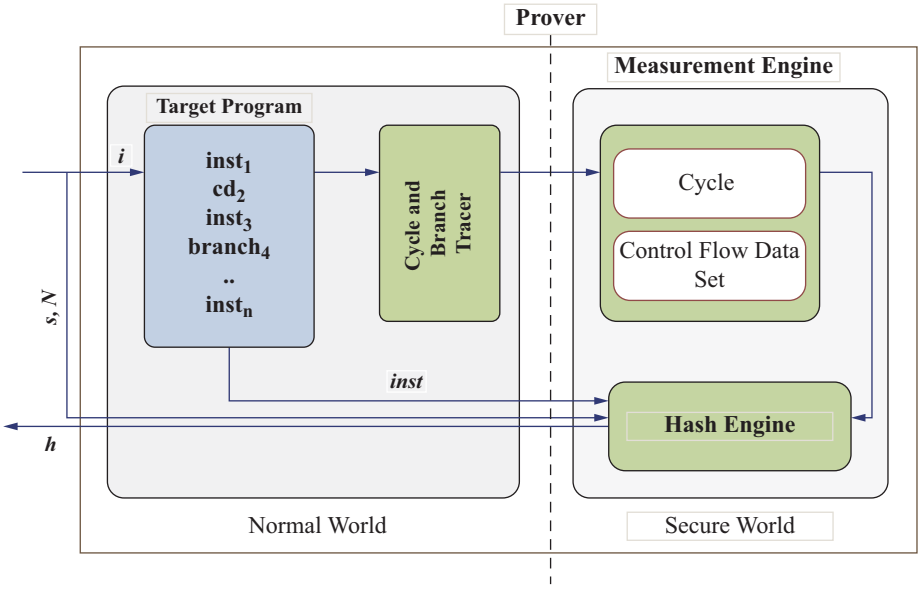


Figure 3: Proof-of-concept implementation.

4.2.1 Evaluation metrics

We use the following three properties to evaluate our approach (see Table 1).

Table 1: Device runtime property data.

Runtime property	Data instances
Number of instructions	1,121
Number of cycles	887
Number of branch instructions	89

We first experimented with the accuracy of the classifier for device attestation using three features: number of instructions, number of cycles, and branches for representative applications. To test against a given embedded application, we treated the class label of a genuine device as 1 and 0 otherwise. As shown in Figure 4, we achieved a high verification rate of ≥99.5%, showing the classifier's robustness. This is essential as the classifier does not generate too many false positives.

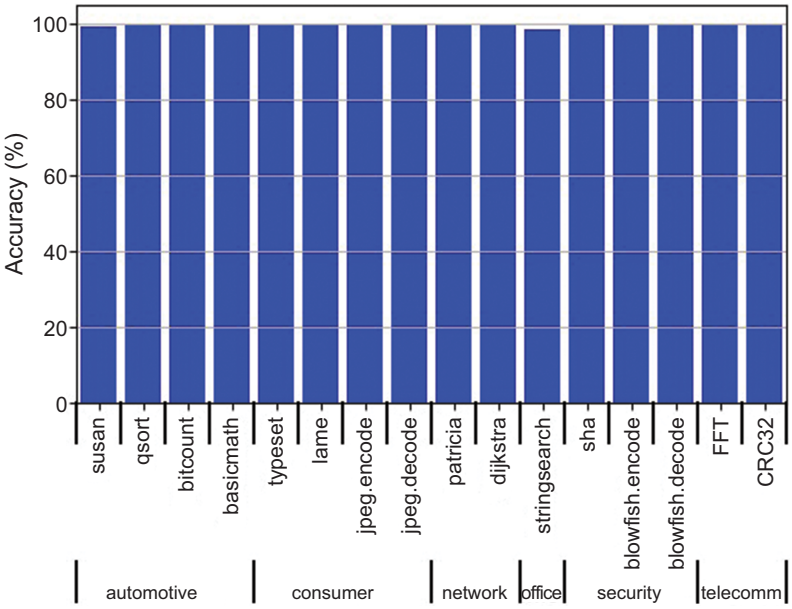


Figure 4: Device verification accuracy for various representative embedded applications.

4.2.2 Runtime performance

Each context switch takes around 7.2 µs. We measure execution time by reading the counter-timer physical count register, CNTPCT_ELO.

We employ the same approach as DADS [37], where $\boldsymbol{O}$ initiates swarm attestation by sending an attestation request to the initiator device, $D1$ (see Figure 5). This request comprises session identifier s, program input i, and Nonce, $\mathbf{N}$. After successful attestation, $D1$ takes the verifier's role and sends an attestation request to its neighbors in a similar fashion. This way, each device in the swarm is attested and then appointed to securely verify others in a decentralized manner (i.e., without $\boldsymbol{O}$ interference). In addition to that, subsequent attestations can be instantiated by any genuine member using the previously given attestation identifier. Here, the

newly assigned initiator device can use a new program input by randomly choosing from the valid program input set already stored in the device.

Figure 6 shows runtime performance of our approach for a single device. The figure shows that attestation overheads look linear to their runtime events. This is because each runtime event causes a context switch and execution of the corresponding attestation-related computation in the *Secure World*.

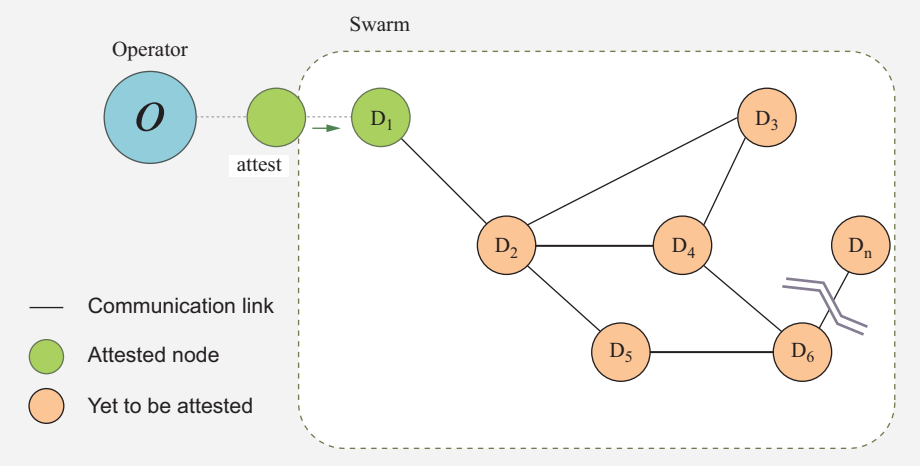


Figure 5: Swarm attestation approach [35].

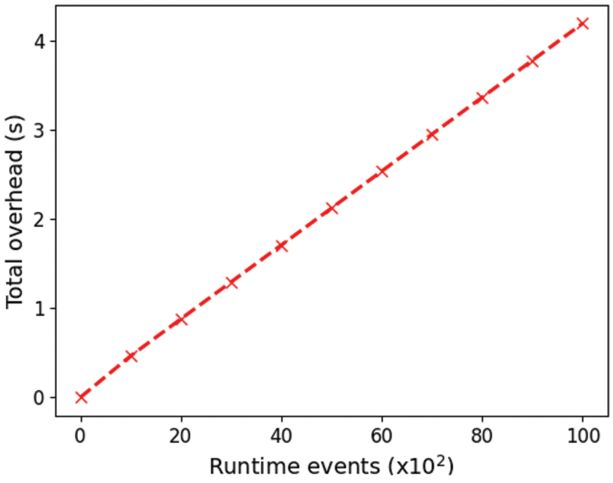
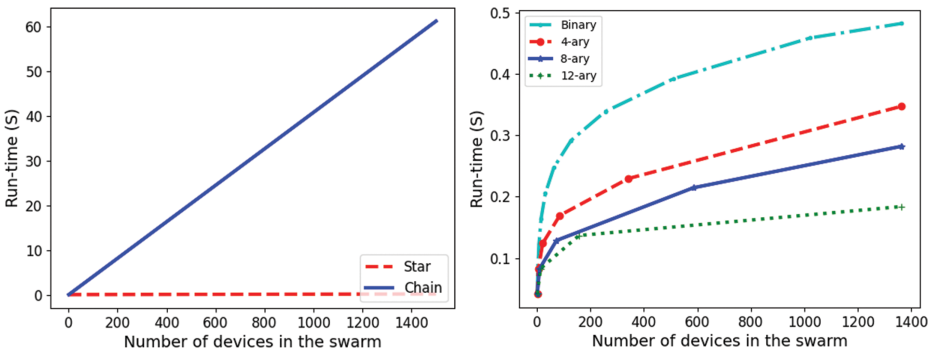


Figure 6: Attestation overhead.

We evaluated the runtime cost of our approach on device swarms (i.e., swarms of connected devices) using OMNeT++ [43] simulation environment. We carried out experiments with chain, star, and tree (i.e., binary, 4-ary, 8-ary, and 12-ary) topologies.

For this, we execute and measure the attestation overhead of a real CPS/IoT application. We chose fast Fourier transform (FFT) application from MiBench suite to carry out experiments as it exhibits a 100% attestation accuracy (see Figure 4). We emulated attestation overheads as delays based on measurements obtained from ARM-TrustZone-based implementation on the OP-TEE.

As shown in Figure 7a, runtime of our approach on FFT for star-and-chain topologies grows linearly, although the star looks constant due to the scale of the graphs. Having multiple connected links means that the star topology allows attestation to be executed in parallel, resulting in better performance. However, star topology implies a general case as all species have one common ancestor.



(a) Star and Chain topologies (b) Tree topologies

Figure 7: Runtime performance of our approach on FFT benchmark for various topologies.

The more realistic topology in a swarm of many devices is a tree. As depicted in Figure 7b, runtimes appear logarithmic to the number of devices in the swarm for various tree topologies. In addition to that, our ML-based approach performs better when the number of child nodes increases.

4.2.3 Computational cost

The major part of the computational cost is due to cryptographic operations, that is, creating and verifying h. Since we utilize a decentralized attestation technique and each swarm member is involved in device verification, every device, D_i, creates $g_i - 1$ and verifies at most p_i MACs, where $p_i \leq (g_i - 1)$, and g_i is the number of neighbors of D_i. Upon successful attestation, prover takes the verifier's responsibility so that it

computes $g-1$ MACs for attestation requests to its neighbors while it verifies at most $g-1$ MACs for verifying their reports.

4.2.4 Communication cost

Our implementation has a MAC size of $lmac = 32$ bytes. We also use $l_N = 20$, $l_s = l_i = 8$, $devID = 4$, MACs = 20, and $h = 32$, where all lengths are given in bytes.

Communication cost depends on the amount of data transferred between devices, which essentially relies on the number of neighbors. In our implementation, attestation requests are 68 bytes while attestation reports are 64 bytes. Consequently, the communication overhead of each device D_i is sending at most $64 + 68w_i$ bytes and receiving at most $64p_i$ bytes, where g_i is the number of D_i's neighbors, w_i is $g_i - 1$, and $p_i \leq g_i - 1$.

4.2.5 Memory requirements

Each device D_i stores four write-protected values: session identifier s, $devID$ for neighbor devices with which it shares a common attestation key, attestation key (K), and a pre-installed hash of the device's attestable memory (c). Thus, the storage overhead for every device, D_i, is estimated as $72 + 4g_i$ bytes, where g_i is the number of neighbors of D_i. In addition to this, D_i should also store valid program inputs along with their corresponding attestation results (i.e., runtime information consisting of a control-flow graph and sequences of operations on the program's critical data). However, one can remove this additional memory requirement by letting the verifier regenerate the runtime information while verifying the prover's report. In general, this additional memory requirement depends on the number of valid program inputs and the complexity of the application to be attested. In general, given the memory size provided as a nonvolatile flash in low-end embedded systems and the availability of low-cost external flash memory cards (i.e., the price of a transflash card with 16 GB is <$6), the memory overhead is within the acceptable range.

4.2.6 Energy cost

Energy is required to send/receive a message, generate a random number, and compute MAC/VERMAC (i.e., h). Let E_{send}, E_{recv}, E_{prng}, and E_h be the energy required to send one byte, to receive one byte, to generate nonce N and to compute runtime attestation result h.

Then the energy consumption for each device D_i is estimated as follows:

$$(D_i) \leq (64 + 68w_i) \cdot E_{\text{send}} + (64p_i) \cdot E_{\text{recv}} + E_{\text{prng}} + (2p_i + w_i + 2) \cdot E_{\text{mac}} \quad (1)$$

where g_i is the number of neighbors of D_i, w_i is $g_i - 1$, $p_i \leq g_i - 1$, and $E_{\text{mac}} = E_h$.

5 Security considerations

Our adversarial model considers remote and local adversaries that pose static and runtime attacks while physical attacks are out of our scope (recall Section 2). The goal is to provide an accurate and fresh attestation report of the application code in the prover, **Pro**.

5.1 Attack threat model perspective

Adversary ***Adv*** may try to mount runtime attack by corrupting memory load and store operations.

Besides, ***Adv*** could also inject some malicious code into the application that may result in the modification of the application's original control flow. In addition to this, it could also try to pose noncontrol data attacks by corrupting data variables. Here, the execution of the application would eventually change the expected control flow. In both cases, the modification is revealed in the resulting control-flow graph. Our approach, thus, detects such attacks that pose malicious control-flow deviations by measuring the resulting control flow.

Further, the new technique detects any changes of the application memory and other attestation-related code used for runtime property collection. Thus, all modifications are revealed in the attestation results.

5.2 Implementation approach perspective

Code performing attestation-related operations (i.e., execution/runtime property collection and measurements) is in a hardware-protected secure memory. Besides, the hardware extensions in the ARM TrustZone guarantee to receive all the runtime information from the processor, and measurements are taking place in a secure execution environment called *Secure World* (i.e., a trust anchor), which means ***Adv*** cannot tamper with the measurement code. Any modification on the running code is revealed on the measured value (attestation report). Suppose that ***Adv*** wants to change the result by forging the hashed attestation report (h) and trick the verfier **Ver**. As ***Adv*** cannot tamper with code measurement entity in **Pro**, ***Adv*** should forge the hashed value, h, if it is going to deceive **Ver**. Given the forgery resistance nature

of MAC (recall that we use MAC based on BLAKE2, see Section 4), the probability that **Adv** forges h is negligible. Besides, attestation key K is read only by attestation code (attestation measurement entity), to which **Adv** cannot tamper with. Thus, it leaks no information regarding the key.

Next, we consider the case where **Adv** fakes an attestation request and masquerades as **Ver** to cause a DoS on **Pro**. Attest Request comprises three basic components: session identifier s, program input i, and nonce N. A single s is used by every member of the swarm for the duration of one attestation session. Obtaining one s means **Adv** can repeatedly send requests by attaching it with possible i and nonce N values. However, N is a randomly generated number while i is selected from valid program input set. In addition to that, all s, i, and N are authenticated via Mac(K; s || i || N). Thus, neither case is feasible in light of key storage and MAC features discussed earlier.

6 Related work

Applications in CPS and IIoT involve interconnected devices that operate in large numbers. RA is generally considered as a viable technique to establish trust between cooperating devices by verifying the swarm members' integrity.

Table 2: Quality-wise comparison of state-of-the-art swarm attestation approaches.

	SEDA [27]	LISA-s [28]	SANA [29]	SAFE^{d} [44]	DADS [37]	This work
Verifier scheme	Centralized	Centralized	Centralized	Decentralized	Decentralized	Decentralized
Attestation approach	Static	Static	Static	Static	Static	Static and runtime (both)
Feature to be attested	Binary	Binary	Binary	Binary	Binary	Runtime property (Ml-based)
Device mobility	No	Partial	No	Partial	Yes	Yes
Supported topology	Static	Partial	Static	Partial	Dynamic	Dynamic
Security against single point of failure	No	No	No	Yes	Yes	Yes

SEDA [27] is the first attempt toward attestation in device swarms. This technique assumes one external entity (***O***) controls the whole attestation process. ***O***, then, generates attestation requests and propagates the same to swarm members. After that, it receives a cumulative member devices' attestation report. SEDA's attestation approach is static in nature (i.e., computes a hash of its attestable memory). This means that it does not capture the target program's runtime behavior. LISA-s [28] builds on top of SEDA and adds up the idea of Quality of Swarm Attestation, providing detailed swarm attestation information such as identifying successfully attested devices along with their total number. Similar to SEDA, LISA-s also employs a centralized (having one **Ver** for the whole swarm) and static attestation approach.

SANA [29] is another static RA attempt in device swarms. Unlike SEDA and LISA-s, this technique utilizes untrusted entities (aggregators) to propagate messages from other aggregators and provers to the central verifier and vice versa.

Toward making swarm attestation scalable and efficient, DADS [37] and SAFE^{d} [44] propose a decentralized approach, where a pair of swarm members validate their integrity without external verifier's interference (i.e., they do not rely on an external verifier). These two approaches remove a single point of failure by allowing swarm members to coordinate and self-protect the underlying network. However, similar to all prior works, they are static. Table 2 compares and summarizes the basic features provided by various swarm attestation schemes.

7 Protocol extensions

7.1 Device recovery

Attestation aims at detecting if devices are infected and/or malfunctioning. Consequently, the compromised devices would need to restore their software to the initial state so as to join the group again and collaborate to achieve the intended goal. In this regard, our ML-based decentralized approach helps swarm members to attest to each other. After that, corrupted devices heal themselves with the correct/updated code by interacting with the honest (attested) neighbor nodes.

7.2 Code update

These smart devices also need their software gets updated to fix bugs and improve functionality from time to time. This issue necessitates system assistance for over-the-air code updates. This decentralized approach can be extended to update the underlying code (i.e., embedded application) on IoT devices.

8 Conclusion

In this chapter, we presented a decentralized swarm attestation scheme that verifies the runtime behaviors of a running program. Our verification approach first integrates the tracking of instructions being executed and control-flow deviations through checking branch instructions. It then uses these collected execution time properties (i.e., the total number of instructions, instruction cycles overserved, and the total number of branch instructions) to train the ML classifier and attest new instances accordingly. We also discussed how this technique could be applied in systems with a large number of embedded devices (device swarms). We demonstrated the feasibility of our approach using ARM TrustZone security extensions for embedded devices. Performance evaluations of our approach reveal that it incurs very low overhead while assuring both static and runtime verifications. We also showed that the new technique provides robust security features.

Notation

Table 3: Variables and parameters.

Entities	
O	Swarm operator/owner
D_1	Swarm attestation initiator (for a specific instance)
D_i	Device i
Ver	Verifier
Pro	Prover
Adv	Adversary
Swarm parameters	
z	Total number of devices in the swarm
g_i	Number of neighbors of D_i
$P \leq g_i - 1$	Number of children of D_i in the attestation tree
Device parameters	
devId	Device identifier
κ_{ij}	Attestation key shared between D_i and D_j
P	Target program P
pf(P)	Property fingerprint of Program P
i	Program input
Protocol parameters	
instTotal	Total number of instructions
cycleTotal	Total number of instruction cycles

Table 3 (continued)

Entities	
branch	No. of branch instructions
s	Session identifier
N	Nonce
T	A Boolean variable indicating attestation success/failure
h	MAC digest of runtime property fingerprint
(*Src*, *Dst*)	Source and destination addresses of an instruction
st_{ab}	Storing instruction a on critical data b

Acronyms

ARM	Advanced RISC machines
CPS	Cyber-physical systems
DoS	Denial of service
IIoT	Industrial Internet of things
IoT	Internet of things
KNN	k-Nearest neighbors
MAC	Message Authentication Code
MCU	Memory controller unit
ML	Machine learning
OP-TEE	Open-source trusted execution environment
PC	Program counter
RA	Remote attestation
ROM	Read-only memory
TEE	Trusted execution environment
VERMAC	Verify Message Authentication Code

References

[1] Ahin, E. S. (2004). Swarm robotics: From sources of inspiration to domains of application. In *International workshop on swarm robotics* (pp. 10–20). Springer.

[2] Valavanis, K. P., & Vachtsevanos, G. J. Eds. (2015). *Handbook of unmanned aerial vehicles* vol. 1. Dordrecht, Springer Netherlands.

[3] Dasgupta, P. (2008). A multiagent swarming system for distributed automatic target recognition using unmanned aerial vehicles. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *38*(3), 549–563.

[4] Gora, P., & Rüb, I. (2016). Traffic models for self-driving connected cars. *Transportation Research Procedia*, *14*, 2207–2216.

[5] Mitchell Waldrop, M. (2015). Autonomous vehicles: No drivers required. *Nature News*, *518*(7537), 20.

[6] Fagnant, D. J., & Kockelman, K. (2015). Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, *77*, 167–181.
[7] Illera, A. G., & Vidal, J. V. (2014). Lights off! the darkness of the smart meters. In *BlackHat Europe*.
[8] Vijayan, J.(2010). Stuxnet renews power grid security concerns. *Computer world* (p. 26). Retrieved February 12, 2022 from *https://www.computerworld.com/article/2519574/stuxnet-renews-power-grid-security-concerns.html*
[9] Kabay, M.(2010). Attacks on power systems: *Hackers malware*. Norwich University. Retrieved February 12, 2022 from *https://www.networkworld.com/article/2217684/attacks-on-power-systems–hackers–malware.html*.
[10] Pollet, J., & Cummins, J. (2010). Electricity for free? The dirty underbelly of SCADA and smart meters. In *Proceedings of Black Hat USA*.
[11] Trautman, L. J., & Ormerod, P. C. (2017). Industrial cyber vulnerabilities: Lessons from Stuxnet and the internet of things. *University of Miami Law Review*, *72*, 761.
[12] Chen, D. D., Woo, M., Brumley, D., & Egele, M. (2016). Towards automated dynamic analysis for Linux-based embedded firmware. *NDSS*, *1*, 1–1.
[13] Costin, A., Zaddach, J., Francillon, A., & Balzarotti, D. (2014). A {large-scale} analysis of the security of embedded Firmwares. In 23rd USENIX Security Symposium (USENIX Security 14) (pp. 95–110).
[14] Sadeghi, A.-R., & Stüble, C. (2004). Property-based attestation for computing platforms: Caring about properties, not mechanisms. In Proceedings of the 2004 workshop on New security paradigms (pp. 67–77).
[15] TCG. Trusted Computing Group(TCG). Retrieved February 12, 2022 from: *http://www.trustedcomputinggroup.org/*.
[16] Parno, B., McCune, J. M., & Perrig, A. (2010). Bootstrapping trust in commodity computers. In Security and privacy (SP), 2010 IEEE symposium on (pp. 414–429). IEEE. *https://doi.org/10.1109/sp.2010.32*.
[17] Seshadri, A., Luk, M., & Perrig, A.(2008). Sake: Software attestation for key establishment in sensor networks. In International Conference on Distributed Computing in Sensor Systems 372–385. Springer. https://doi.org/10.1007/978-3-540-69170-9_25.
[18] Seshadri, A., Perrig, A., Leendert, V. D., & Khosla, P.(2004). Swatt: Software-based attestation for embedded devices. In null (p. 272). IEEE.
[19] Armknecht, F., Sadeghi, A.-R., Schulz, S., & Wachsmann, C. (2013). A security framework for the analysis and design of software attestation. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (pp. 1–12). ACM. https://doi.org/10.1145/2508859.2516650.
[20] Eldefrawy, K., & Tsudik, G. (2012). Aur´elien Francillon, and Daniele Perito. Smart: Secure and minimal architecture for (establishing dynamic) root of trust. *NDSS*, *12*, 1–15.
[21] Koeberl, P., Schulz, S., Sadeghi, A.-R., & Varadharajan, V. (2014) Trustlite: A security architecture for tiny embedded devices. In Proceedings of the Ninth European Conference on Computer Systems (p. 10). ACM.
[22] Brasser, F., Mahjoub, B. E., Sadeghi, A.-R., Wachsmann, C., & Koeberl, P. (2015). Tytan: Tiny trust anchor for tiny devices. In Proceedings of the 52nd Annual Design Automation Conference (p. 34). ACM.
[23] Sasaki, T., Tomita, K., Hayaki, Y., Liew, S. P., & Yamagaki, N. (2020). Secure IoT device architecture using TrustZone. In 2020 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops) (1–6). IEEE.
[24] ARM Holdings. (2009). Arm security technology: Building a secure system using TrustZone technology. https://developer.arm.com/documentation/genc009492/latest.

[25] McKeen, F., Alexandrovich, I., Anati, I., Caspi, D., Johnson, S., Leslie-Hurd, R., & Rozas, C. (2016). Intel® Software Guard Extensions (Intel® SGX) support for dynamic memory management inside an enclave. In Proceedings of the Hardware and Architectural Support for Security and Privacy 2016 (pp. 1–9).
[26] Viega, J., & Thompson, H. (2012). The state of embedded-device security (spoiler alert: It's bad). *IEEE Security & Privacy*, *10*(5), 68–70.
[27] Asokan, N., Brasser, F., Ibrahim, A., Sadeghi, A.-R., Schunter, M., Tsudik, G., & Wachsmann, C. (2015). SEDA: Scalable embedded device attestation. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (pp. 964–975). ACM.
[28] Carpent, X., ElDefrawy, K., Rattanavipanon, N., & Tsudik, G. (2017). Lightweight swarm attestation: A tale of two LISA-s. In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. ACM.
[29] Ambrosin, M., Conti, M., Ibrahim, A., Neven, G., Sadeghi, A.-R., & Schunter, M. (2016). Sana: Secure and scalable aggregate network attestation. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 731–742). ACM.
[30] Abera, T., Asokan, N., Davi, L., Ekberg, J.-E., Nyman, T., Paverd, A., Sadeghi, A.-R., & Tsudik, G. (2016). C-flat: Control-flow attestation for embedded systems software. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 743–754). ACM.
[31] Earnshaw, R. (2003 October) Procedure call standard for the arm architecture. *ARM Limited*. http://www.0x04.net/doc/elf/psABI-arm-call.pdf.
[32] Guthaus, M. R., Pingenberg, J. S., Austin, T. M., Mudge, T., & Brown-mibench, R. B. (2001). A free, commercially representative embedded benchmark suite, wwc-4. *IEEE International Workshop on Workload Characterization* 2001 (Vol. 10) pp. 3–14.
[33] Bench Mark. MiBench Version 1.0: Retrieved February 12, 2022 from http://vhosts.eecs.umich.edu/mibench/.
[34] Soucy, P., & Mineau, G. W. (2001). A simple kNN algorithm for text categorization. In Proceedings IEEE International Conference Data Mining (ICDM) (pp. 647–648).
[35] Ali, N., Neagu, D., & Trundle, P. (2019). Evaluation of k-nearest neighbour classifier performance for heterogeneous data sets. *SN Applied Sciences*, *1*(12), 1–15.
[36] Taylor, B., Marco, V. S., Wolff, W., Elkhatib, Y., & Wang, Z. (2018). Adaptive deep learning model selection on embedded systems. *ACM SIGPLAN Notices*, *53*(6), 31–43.
[37] Wedaj, S., Paul, K., & Ribeiro, V. J. (2019, July 16). DADS: Decentralized attestation for device swarms. *ACM Transactions on Privacy and Security (TOPS)*, *22*(3), 1–29. https://dl.acm.org/doi/pdf/10.1145/3325822
[38] Lynnette Reese Mouser Electronics. Website: Powerful SoCs Turbocharge Wearables' Future. Retrieved February 12, 2022 from https://www.bisinfotech.com/powerful-socs-turbocharge-wearables-future/.
[39] OP-TEE: Open source implementation of TEE. (last accessed February 12, 2022). https://www.op-tee.org/
[40] McGillion, B., Dettenborn, T., Nyman, T., & Asokan, N. 2015. Open-TEE–An Open Virtual Trusted Execution Environment. In 2015 IEEE Trustcom/BigDataSE/ISPA Vol. 1 (pp. 400–407). IEEE.
[41] BLAKE2: hash function BLAKE2. (last accessed February 12, 2022.). https://blake2.net/
[42] Aumasson, J.-P., Neves, S., Wilcox-O'Hearn, Z., & Winnerlein, C. (2013). BLAKE2: Simpler, smaller, fast as MD5. In International Conference on Applied Cryptography and Network Security (pp.119–135). Springer.
[43] OpenSim Ltd. OMNeT++ discrete event simulator. (last accessed February 12, 2022). http://omnetpp.org/.
[44] Visintin, A., Toffalini, F., Conti, M., & Zhou, J. Safe^{d}: Self- attestation for networks of heterogeneous embedded devices, 2019. https://arxiv.org/ abs/1909.08168.

Sangeeta Mittal

A review of machine learning techniques in cybersecurity and research opportunities

Abstract: The past few years have seen a dramatic surge in Internet usage of all individuals. Moreover, this usage has evolved from casual surfing to serious online businesses. Pandemic has further greased the wheels of cyber presence as many organizations plan to go completely online, thereby increasing the importance of cybersecurity than never before. Cyber monitoring systems generate humongous amount of data that is difficult to be analyzed manually and thus machine learning (ML) techniques have been leveraged to make sense out of this data in real time to prevent cyberattacks. A systematic summary of ML applications in cyber defense have been provided by mapping threat categories to ML-based solutions. On examining these implementations, it has been concluded that issues like availability of real-world attack data, concept drift in normal behavior, and processing power issues are still not resolved. It has been discussed that due to these prevailing issues, the performance of ML-based security systems may decrease dramatically in actual implementations. New research directions in this domain have also been highlighted in the chapter.

Keywords: Cyber threats, Cyber Defense, Advesarial Learning, Federated Machine Learning, Insider Threats, Denial-of-Service

1 Introduction

Cybersecurity encompasses mechanisms for ensuring confidentiality, integrity, and availability of information during access, transit, and storage [1]. In the context of this work, cybersecurity is considered to be an umbrella term to represent techniques for information, network, Internet, and computer system security. Cybercrime is any unlawful act that can lead to failure, damage, error, accidents, harm, or any other undesirable event in cyberspace. Protagonists are continually being challenged to thwart cybercrimes and protect critical infrastructure and information [2]. A sound and complete security system is a myth as there is not any single known system that finds and thwarts all possible attacks and does not give any false alarms. Moreover, as organizations are profit oriented, it is neither practical nor desirable to address all the attack possibilities. As a result, attacks like denial-of-service (DoS), spam, malware, and phishing attacks are still giving a hard time to cybersecurity providers and researchers

Sangeeta Mittal, Jaypee Institute of Information Technology, India,
e-mail: sangeeta.mittal@jiit.ac.in

https://doi.org/10.1515/9783110766745-005

[3]. Owing to a sudden surge in remotely working employees in every sector, there has been a manifold increase in the number of these attacks since the pandemic era.

To trade off the cost incurred in securing assets and attacks thwarted, some effective prioritization has to be done. One way to do this is to assess the context in which these assets are being accessed and utilized. For example, sudden heavy traffic on a website may be a DoS attack as well as legitimate surge in demand due to some external event, which acts as the context in which demand has increased. As a result of multilevel monitoring and logging, a huge amount of data gets accumulated at a very high rate [4]. A major challenge is thus to process this data in real time to mine relevant threat intelligence in real time for immediate action as well as long-term system hardening. Machine learning (ML) techniques have proven to be great at mining information from heaps of data. Reality mining can correlate various events across different departments, and timely predict/detect the onset of attack and most of the time thwarts it completely [5].

Gartner's recent report on top security trends mentions cyber-savvy board members who care a lot about cybersecurity and are increasing chief information security officer support, budget, resources, as well as expectations [6]. Also, in security operation centers, integrated incident response and threat intelligence are being added [7]. This paradigm shift underlines more prominently the importance of use of ML in cybersecurity. ML-based security solutions applied in cybersecurity and their pitfalls have been explored in this work.

2 Threats to cybersecurity and defense strategies

2.1 Threats to cybersecurity

Cybercrimes have increased manifold in Covid-19 era due to a sudden increase in remote workforce and lack of readiness of organizations against cyberthreats [8]. Reports say that a cyberattack attempt occurs every 39 s. Thus, any 24 × 7 online device sustains about 2,200 cyberattacks every day [9]. Attacks are becoming sophisticated day by day and attackers are always a step ahead of defenders to develop novel attacking mechanisms. However, the most common types of attacks, shown in Figure 1, are still being used to exploit the vulnerabilities in newer ways [10, 11].

Illegal access to privileged systems and accounts is one of the most common types of attacks. This includes password guessing attacks and intrusion attempts to break in the system. Another category of attacks that is being practiced a lot these days are identity and information theft attacks. Identity thefts lead to impersonation and can be done by various phishing variants like smishing, vishing, and spam emails. Information theft is possible by sniffing and side-channel analysis. Attacks on integrity of data are also critical as the success of the financial Internet is based

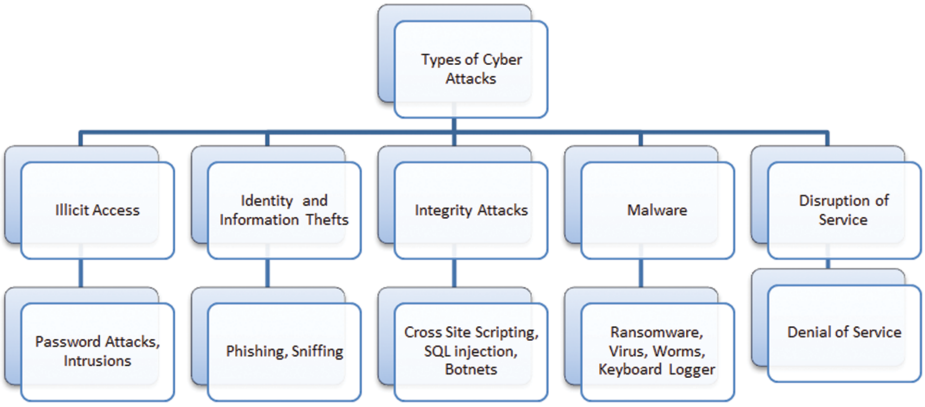


Figure 1: Common types of cyberattacks.

on the sanctity of transactions. Man-in-the-middle-based attacks like cross-site scripting and SQL injection using botnets are common types of attacks on data integrity. Malware is an intentionally written malicious software that intends to harm the computer in which it gets installed as well as tries to spread itself to as many other machines as possible. Software-based attacks can also be carried out by unintentional insecure code that carries vulnerabilities to be exploited by attackers. However, malware written for specific purposes causes greater loss as has been evident through real-world examples of ransomware, virus, and worm attacks. Majority of the attacks target availability of a system of public interest leading to its degraded performance and deny access by legitimate users. Table 1 lists the attack vector used in these attacks to enhance their effectiveness and coverage.

Table 1: Means of launching attacks.

Attack vector	Type of attack
Social engineering, email spam	Identity and information thefts
Drive-by-download, P2P networks, free software sites	Spreading malware
Network based	Intrusions and disruption of service
Buffer overflow	Illicit access
System software (operating system, database management system, browser, etc.) flaws	Integrity attack

Method–opportunity–motivation is a common paradigm to summarize cyberattacks. While methods have been discussed as types of attacks, motivation is financial gain in a large number of cases. The opportunity or means of launching various attacks is given in Table 1. Social engineering is a popular method to launch identity and information theft

attacks, such that the user divulges sensitive information to perpetrators. Malware is hosted in freeware sites and P2P networks nowadays while in earlier times it was usually through flash drives and directly connected machines. Network-based access is mandatory for remote access or Internet-based usage of servers. It is very difficult to automate identification of malicious network traffic from benign traffic and attackers use this access vector to intrude into systems or launch DoS attacks. Buffer overflows are unavoidable unintentional vulnerabilities in application source code. It has been exploited for code injection and taking administrative controls of target system. Other operating systems and database management system flaws cause system states and data to be modified illegally, thus making it inaccessible.

2.2 Defense strategies for cybersecurity

Ensuring cybersecurity is a humongous task. A typical day in any IT-based organization is receiving and sending data through publicly shared networks, remote logins from its employees and other stakeholders into the company servers, end users interacting with organization's servers to fetch data and e-mail-based communication. All these sets of events occur at the rate of several times per second in a medium organization, thus creating big datasets to be handled.

Apart from this, a lot of log data from systems, networks, and users is collected continuously. Defense mechanisms can be developed by experts having domain knowledge and also by using this data. Strategies employed for cybersecurity can be categorized as methods developed for cybersecurity analytics, prevention, and detection [12]. Analytics is targeted toward getting deeper insights into attacks and their root cause analysis. Prevention techniques are a set of proactive strategies that help prevent attacks in the first place to avert any harm caused to systems. Occasional attacks on systems are imminent even though security measures may be in place. In such scenarios, it is important to detect the attack as soon as possible so that harm inflicted can be minimized. Thus, detection, response, and recovery strategies of securing systems are also very important.

Each category of major attacks and threats to the system can be handled in all three ways and levels. Data available at different platforms and from different systems is utilized for offline statistical analytics to create attack detection engines. This engine can be embedded into online scanning techniques for real-time attack prevention, and if needed, detection and response.

In Table 2, for each of the prominent attacks, all three possible defense strategies have been listed. For example, for handling the menace of malicious software, offline analytic methods can be applied on available malicious codes and their execution behavior to design anti-malware solutions. For prevention, this software can be utilized for scanning real-time traffic for the presence of malware before it enters the system. New malware can still remain undetected, and thus, enter the system.

Table 2: Defense strategies.

	Analytics (offline)	**Prevention (online)**	**Detection and response (online)**
Illicit access	Password strength indicator	Hashed password storage	Password compromise detection via user behavior detection
	Web application intrusion analytics	Anti-phishing	Script detection in Website code
	Breach and attack simulation (BAS)	Penetration testing	Insider threat/ advanced persistent threat detection
	Botnet and DDoS characterization	Intrusion prevention system	Intrusion detection system
Identity/ information theft Phishing	Algorithms for secure biometrics	Biometric identity verification	Replay attack and fake template detection
	Generalized masquerader profiling	Identity theft prevention	Identity theft/ masquerade detection
	Characterization of side-channel leaks	Information leakage prevention	Man-in-the-middle and side-channel attack detection
	Featurization of malicious URLs and phishing pages	Real-time alerts	Phishing detection
Integrity attack	Software vulnerability enumerators	Fuzzing	Detection of undesired runtime behavior
	Featurization for file-type distinction	File-type identification	Detection of masqueraded file from behavior
	Website vulnerability analytics	Detecting code and command injection vulnerabilities and remote code execution possibilities	Detecting unauthorized record updates
Malware	Malware analytics, anti-malware design	Malware scanners	Static and dynamic malware detection and quarantine
Disruption of service	Featurize intrusions and disruptions	Intrusion prevention systems	Network anomaly detection

In such cases, detection and response strategies should take over to find and quarantine the malware as soon as possible. Existing literature has been well examined in this work to discuss ML-based solutions for each type of defense strategies and their effectiveness.

3 Usage of machine learning in defense strategy implementations

ML is a set of soft computing techniques that allow decision models to be built upon knowledge extracted from large labeled/unlabeled datasets [13]. Depending on the underlying information available and classification task at hand, clustering, class prediction, and regression models can be developed. The process of ML-based defense involves steps shown in Figure 2.

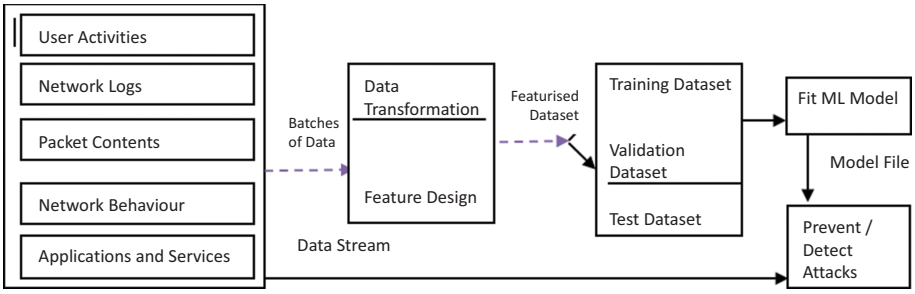


Figure 2: Flow of machine learning-based defense.

Data are collected from source and are used for training and testing. Data from the training set is used to build the model. Input data have to be cleaned, transformed, and sometimes normalized to make them suitable for model consumption. Transformations may involve one or more steps like normalization, discretization, and standardization. A plethora of ML techniques are available to learn classification and regression models from multivariate time series images, text, and numerical data. Performance of an ML model is dependent on extraction of relevant features that are extracted from data before applying ML. Training of models is done on the chosen dataset, followed by validation and testing, and finally deployment in the real world.

In the cybersecurity domain, ML techniques can be applied on data from previous cyberattacks and develop models to recognize patterns of normal and abnormal system behavior, for example, pattern-based detection finds applications in spam detection, malware detection, and botnet detection. In some other cases, where the

patterns of malicious activities are not known, the attack detection methods are finding deviations from normal and thus termed as anomalies.

Like any security system, cybersecurity solutions are also meant to prevent attacks. For this, attack information from experts and previous incidents is proactively collected to do analytics and convert the information to intelligence that can be utilized for attack prevention in real time. Robust analytics go a long way in averting future known and unknown threats. In situations where prevention fails and attack happens, detection and recovery mechanisms are to be put in place.

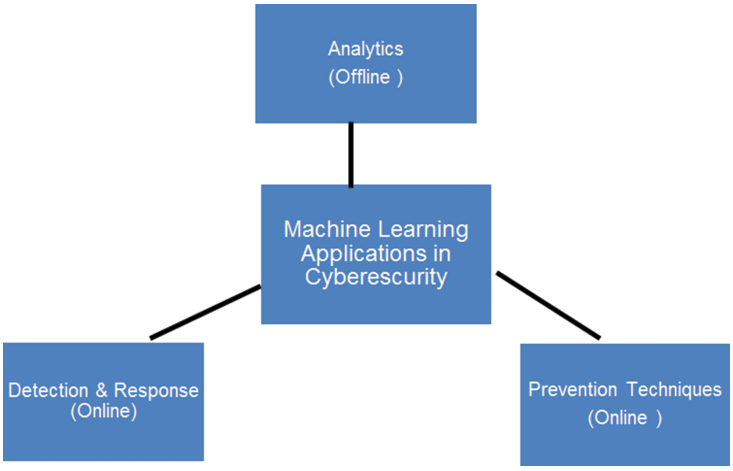


Figure 3: Scope of machine learning in cybersecurity defense mechanisms.

Scope of ML in implementation of robust cybersecurity defense mechanisms is in three possible areas as shown in Figure 3 [14–16].

ML for analytics in cybersecurity: In cybersecurity, data-backed proactive threat detection can be done by employing analytic techniques. It works toward converting raw data into actionable intelligence. Source of data for analytics is very diverse and can range from network traffic, endpoint systems, user behavior data, applications, identity and access management systems, and threat intelligence sources.

ML for prevention in cybersecurity: ML techniques have proved to be very good at intelligence mining from such huge data and obtaining a data-backed model that can be put in place for thwarting attack attempts.

Prevention techniques leverage the intelligence gained from offline analytics to prevent any untoward incident in real time. For prevention of any attack on the IT infrastructure, all inbound data have to be scanned for possible malware, and all outbound data have to be evaluated for a possible bot interaction. Similarly, every remote login attempt has to be evaluated for a compromised password. For thousands of mails received daily, phishing attempts have to be kept on bay. Servers are

to be kept usable and serve legitimate users without falling into the trap of DoS attacks. On top of these requirements, any attack prevention attempt has to be completed in real time without compromising the efficiency of the tasks at hand.

Examples of prevention techniques include firewalls, intrusion prevention systems, identity threat prevention methods, antimalware tools, penetration testing of software, encryption, and most importantly employee awareness. Many of the mentioned technologies make use of ML methods [17]. Hardware features can help improve fuzzing which is soft computing-based solution for automatically detecting vulnerabilities [18]. Babiker et al. [19] discussed application attack prevention techniques like honeypots, firewalls, and intrusion detection systems for real-time detection of attacks on web apps.

Robust analytics and prevention mechanisms help prepare for attacks proactively before they occur. However, attacks are inevitable even if a high level of security is applied. The best thing that can be done in such a case is to detect the attacks at the earliest so that the damage can be minimized.

ML for detection and response in cybersecurity: ML techniques can help learn and detect reliable attack indicators. Detection and response mechanisms need to be aimed at detecting attacks at the earliest, correctly and quickly estimate the scope, scale and impact of the attack, identify the exact technical cause of the attack and eliminate it, and find all possible ways to contain the attack and stop it from spreading further.

This is the most difficult part of cybersecurity handling as it requires combining information from usage logs, packet information, network behavior, user behavior, identity, and fraud data from across various domains. This information has to be linked with each other in real time to detect known and unknown attacks. Offline behavioral analytics learnt earlier can be leveraged to detect and respond to threats in real time. In wake of an attack, its full scope and impact can be assessed by correlating information obtained from status of assets, user and entity behavior, and threat intelligence. Once an attack is detected, actions are to be taken to minimize its effects. However, most of the methods do not study both detection and recovery.

4 Attack-wise ML-based defense strategies

4.1 Illicit access

Illicit access to a resource can be had by either fraudulently obtaining credentials or by bypassing the credential verification mechanism. Phishing is a popularly used method to make the authorized user divulge his/her credentials unknowingly. One of the ways to phish a target is by malicious sites.

Phishing: Phishing URL detection is an application area where simple rule-based checks do not work correctly due to frequent domain change by attackers. Thus, a

lot of ML-based solutions have been proposed for this. Support vector machine (SVM), logistic regression, naïve Bayes, and decision trees are some popularly used supervised algorithms for batch learning of phishing URLs. Jain and Gupta [20] extracted URL-based features, fake login detection, hyperlinks, copied Cascaded Style Sheets (CSS), and fake web identity features (from copyright, favicon, and identity keywords) to apply supervised ML techniques for characterizing phishing websites. With various supervised classification methods being used, they achieved a true positive rate of nearly 100% when a complete set of features were used.

Besides this, first-order and second-order online learning have also been applied. Unsupervised learning and visual similarity ML methods have also been tried to the domain with some success [21]. Overall, it has been observed that online learning methods are more successful and can be improved further for efficiency and scalability. A lot of features can be extracted from the contents and URL of a web page. ML-based solutions also find applications in selecting relevant features for more accurate classification of web pages [21].

In [22], a number of supervised algorithms along with NLP-based featurization have been utilized on a large balanced dataset of phishing and benign data to propose an online anti-phishing system. It was concluded that a classifier based on random forest supervised algorithm performed best while among features instead of statistical features, NLP ones proved effective in detection.

One of the main contributions of authors in this work is to create a recent and big database of phishing and normal sites named, Ebbu2017 Phishing dataset. It lists about 73k URLs in which 36.4k are of legitimate websites while remaining are phishing ones, thus making it a balanced dataset. Seven supervised classification approaches were compared for their detection performance. Random forest and decision trees gave highest accuracy. Similar to earlier works, here also NLP features and hybrid of NLP and word-based features gave better performance than only word-based features.

Sahoo et al. [21] did extensive feature engineering and evaluated existing models of ML. However, the training dataset is of limited size, consisting of 2,141 phishing and 1,918 legitimate websites. Random forest approach once again outperformed all other supervised ML algorithms applied to this dataset in terms of True Positive Rate (TPR) and accuracy.

Spam mails: Unsolicited bulk emails are another problem whose effects can range from being a mere nuisance to being used as a vector for luring the target to reveal their sensitive information. Many ML-based algorithms have been effectively applied to this problem also [23]. Image spam detection is used to extract spam text hidden by spammers in images to evade filters looking for suspicious text directly. Deep learning-based methods have been quite useful in detection of image spam [24].

Web vulnerability: SQL injection vulnerability in web pages that allow users to interact with its servers continues to exist for more than three decades. Two class SVM was used to predict SQL injection vulnerability in web requests [25]. Web server

vulnerability scanner using ML is based on a labeled dataset of web server traffic features and ground truth labels. An ML-based tool for gathering information of products installed on a web server, namely GyoiThon, is based on artificial intelligence techniques [26]. Attack modules to target product-specific vulnerabilities can also be executed. The tool can launch multivector information collection as it has embedded web crawler, APIs for Google search engine, default content explorer, and cloud service analyzer.

Table 3: Features extracted for insider threat detection.

Type of input	Features extracted
Access logs	Files accessed, frequency of access, system access – duration and time
Network logs	Traffic related to DNS, authentication, firewall, proxy
Application logs	Application-specific logs like browser and emails
Chat logs	Prohibited keyword analysis from chat applications
Antivirus logs	IDS/antivirus alarm or warning generation – frequency and severity
User activity monitoring module	Bandwidth consumed per unit of time, login failures, and ping activities

Insider threat detection: Insider threat from disgruntled employees is a growing challenge as it has all the potential to launch advanced persistent threats (APT). Prohibited actions like illicit data transfer and damaging or misuse of organizational resources can be more easily done by insiders rather than an outside attacker. Developing relevant features to identify insider threat detection in real time is a very challenging task. A plausible set of features, as shown in Table 3, have been proposed in [27].

In order to apply ML algorithms, a robust dataset of examples of insider threat indicators has to be developed. One such dataset has been made available by SEI CMU [28]. The dataset has been created synthetically by enacting publicly available case studies of insider threats and some real-world normal data. Isolation forest has been used to detect anomalies in each user's behavior to find the culprit. Even though such a classifier may not be able to prevent attacks, it can be of great help in informing analysts about plausible threats.

Le and Zincir-Heywood [29] worked on CERT insider threat datasets that are already available for assessing insider threat detection methods. User's features for developing ML models were designed at three levels to tell apart genuine and threatening user behaviors. A set of 221 features described user's session characteristics while 824 and 1,092 features defined daily and weekly behaviors, respectively. Usage of metrics like detection rate and delay (in seconds) justifies accurate and

timely detection. Logistic regression, random forest, and artificial neural network (ANN) classifiers were also used.

4.2 Identity and information theft

Identifying theft remains one of the most reported frauds in recent times [30]. Digital ID may get stolen via phishing, vishing, credit card skimming, public Wi-Fi hotspots, and unintentional username and password exposure. Sometimes data breach attacks leave personal information of millions of users exposed on the Internet or sold on the dark web. The Equifax data breach in the USA and Domino's data breach in India are recent examples where personal information of millions of customers was made publicly available [31].

ML techniques have been useful for preventing identity thefts in several proactive ways. Passwords are one of the most popularly used methods for authenticating users remotely. This is based on "What you know" principle and is often combined with other factors like "What you have" also. Guessing attacks are most common attacks on passwords that can be thwarted by keeping strong passwords that are not easily guessable. ML techniques have been used for ensuring that the password is prone to guessing attacks or not [32]. This involves a password strength estimator backed by ML. Many cracking tools check passwords by searching heuristically in password space. Apart from dictionaries, various password cracking tools also create and generate more probable passwords by using contextual information like age, date of birth, and family member name. Deep learning-based PassGAN, proposed in [33], makes use of character distribution and placement in actual passwords to train a generative adversarial network (GAN) and as a result output realistic guesses to crack passwords with higher chance.

Generator and discriminator architecture of PassGAN is given in Figure 4. Discriminator processes passwords in training dataset and those produced by generators through residual networks. Based on the discriminator's feedback, generator adjusts its parameters to output samples in distribution similar to actual passwords. After many iterations and epochs of training, the generator converges and can be used to generate real password guesses. Network administrators can run password files through PassGAN and all users whose passwords are guessed correctly can be prompted for a password change.

Compromised passwords and other sensitive information generally end up being put for sale on the dark web [34]. Tor network is one of the largest networks on the unsearchable web. Tor traffic can be traced by using right contextual information. Deanonymizing Tor has been done using ML techniques by collecting features and information from individual sessions to be able to identify the activity of anonymous users. Training data can be collected by visiting various websites [35]. To uncover activities on the dark web, conmarap/website-fingerprinting repositories are very helpful.

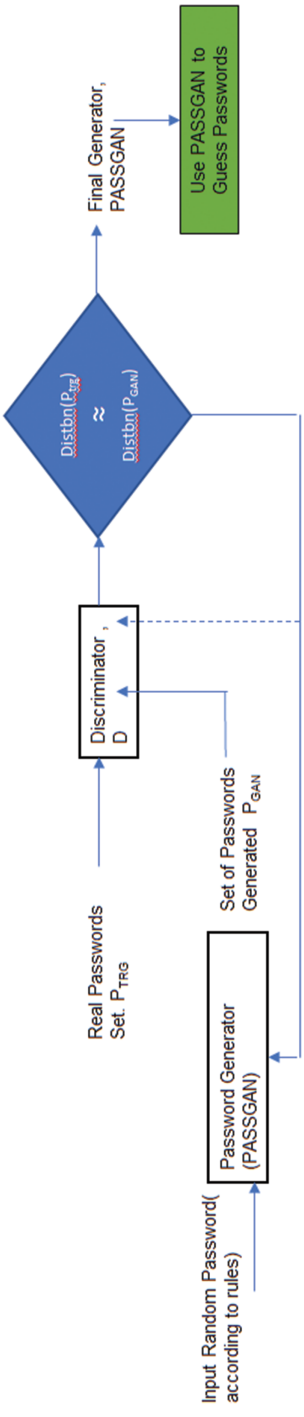


Figure 4: PassGAN architecture.

Penetration testing: Metasploit is a popular tool for penetration testing. Recently, it has been updated to its deep learning-based version, DeepExploit. The underlying improvements are achieved by the use of reinforcement learning while performing deep penetration testing. The improved model does highly efficient automated penetration testing to uncover all vulnerabilities in individual systems and networks. ML solutions have been used to detect sophisticated targeted attacks known as APT [36]. In APT, multiple vectors of attack like social engineering for basic info, spear phishing for targeting high authorities, and drive by download for spreading malware are used. A five-stage ML-based approach to APT detection has been proposed in [36]. ML model has been applied at every stage instead of only in the end as suggested by previous works. In this way, even if the attack is not detected at the first stage, there is a high chance of it being detected at a later stage.

4.2.1 Preventing identity theft by modeling keystroke dynamics

Interestingly, the speed and rhythm of typing can be explicitly profiled for an individual to such an extent that it can become the digital identity of each user. A robust keystroke analyzer based on ML can prove to be a good authentication system [37]. It can enable password-less access as well as detect identity theft quickly. As shown in Figure 5, two time-based features are mainly used for capturing keystroke dynamics. First parameter called dwelling time is actually the keypress duration. The time elapsed between two consecutive keypress and key-release is the second feature and is termed as flight time.

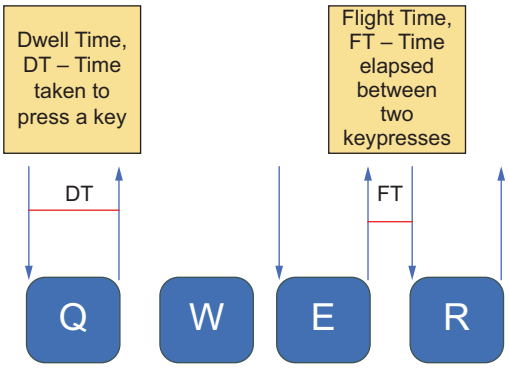


Figure 5: Flight and dwell time features in keystroke biometrics.

For applying ML, Singh et al. [37] used 31 features capturing keypress and release combinations of all keys in a password. They tested KNN, SVC(RBF), random forest, and XGBoost classifier for identifying users from chosen feature sets. It was

found that XGBoost performed very well as compared to other classifiers. However, there is a pitfall that the ML-based system has to be trained for each individual legal user of the system.

4.3 Integrity attacks

Attack on integrity is one of the most dangerous attacks, as it involves not only access of data illegally but also modifying the content and in some cases encrypting it for ransom.

Ransomware detection and prevention: ML and DL techniques are helpful in combating the menace of ransomware. By using behavioral, static, and network features, ML techniques have been able to achieve 78–100% detection rate [38]. However, for such attacks in newer types of networks like Internet of things (IoT) ML-based solutions cannot be applied due to the lack of existing examples.

Another important vector to compromise integrity is through existing software vulnerabilities. Detecting and plugging these weaknesses will go a long way in reducing integrity attacks.

Software vulnerability discovery is considered to be the time taken for the software development itself. Software vulnerability analysis is generally done by techniques such as software penetration testing, fuzz testing, and tainted flow testing [39]. ML has been applied to develop prediction models for software vulnerability by applying methods of anomaly detection and pattern recognition on the underlying code. Unintentional software errors, like buffer overflow, have been the source of major cyberattacks in the past. Deep learning-based system has been proposed to automatically detect such issues in the underlying legacy code especially for C and C++. This is cheaper and more effective than a software tester manually looking for these unintentional flaws. Deep learning finds a useful application here as it is difficult to extract meaningful features that accurately represent erroneous and nonerroneous code. DL techniques automatically extract features from training data. VulDeePecker is a deep learning-based tool to detect buffer overflow vulnerabilities. Tools like code gadgets can be used to make code-based input data applicable to deep learning models like bidirectional long short-term memory (LSTM).

Software testing has been a highly manual process, and thus human expertise and knowledge of software becomes a limiting factor. Fuzzing is an important tool to automate the software testing process in which a large number of inputs are automatically generated to test if any of these cause the program to crash. ML-based fuzzing methods that reduce the probability of retesting similar inputs have recently been developed. NEUZZ, based on neural networks, is one such tool that can be used to find inputs that cause programs to crash with greater probability [40].

Apart from above, there is another aspect of integrity with respect to doctored videos, manipulated text, reviews, and news. Use of deep learning can produce

astonishingly unrecognizable fake images. DeepFake recognition is an emerging technology for detecting videos tampered using DL [41]. ML and DL techniques have been used for fake reviews as well as fake news detection [42, 43].

4.4 Malware

Malware, a short form of "Malicious Software," is any piece of code written with malicious intent. Static and dynamic malware analyses are the two methods of malware detection involving inspection of code and its behavior, respectively. Static analysis involves scanning file contents and reverse engineering of executables to understand its logic. Newer malware can evade being detected statically using obfuscated or encrypted malicious code. In dynamic analysis of malware, suspicious code is executed in a protected environment called sandbox and its behavior-like order of system and API calls is observed. Figure 6 summarizes the scope of ML in malware analysis [44].

In both static and dynamic detection mechanisms, ML-based solutions can be utilized. Detecting known malware can be considered as a classification problem using supervised learning while unknown malware can be detected by clustering algorithms for unsupervised ML.

4.4.1 File-type identification

Executable malware is often distributed as either windows executables (PE32 files) or linux executables known as ELF files. As it is straightforward for antimalware programs to ban these file types, perpetrators normally tend to change the file extension to some other types of allowed files like .js and .pdf. Singh and Singh [45] presented an overview of application of ML to identify that the file contents match the file extension. C4.5 and k-NN performed better than many other ML algorithms when applied on one benign and two malicious datasets of executable files.

4.4.2 Detecting mobile phone, IoT, and other malware

Mobile malware can be distinguished from benign apps by considering the amount of traffic, battery usage, and frequency of connection to the outside world. Given meaningful features, all popular ML algorithms can be used to build classification models. Crowdsourcing can be a great way to make ML algorithms robust. Users can share the system calls made by their application with a central server followed by statistical mining techniques to characterize malware. Features such as CPU usage, memory usage, network traffic, and battery usage can be used to detect mobile phone malware. Android malware detection can also utilize permissions sought

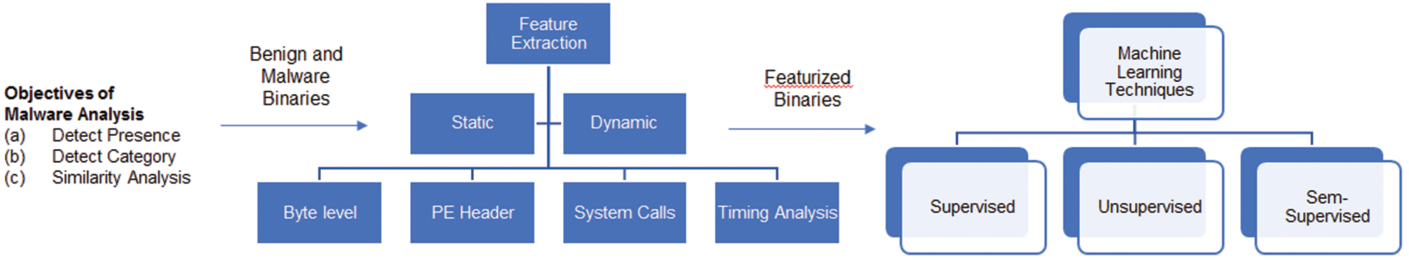


Figure 6: Use of ML techniques for malware analysis.

by apps as features followed by application of ML algorithms for binary classification to goodware and malware as well as multiclassifier to the type of malware [46]. Permissions are also extrapolated by looking at API names, their packages, and parameters used. Malware written often uses simple encryption techniques to avoid exact matching of API names. Malware modeling can also be done as time series interactions using hidden Markov model [47]. Uncorrelated system calls can be used to detect malicious activities. Thus, from the literature it can be concluded that offline analytics using ML is a fast and reliable mechanism to learn classification models to automate online prevention of many types of attack attempts.

4.5 Disruption of service

DoS attacks target the availability of systems to its legitimate users. Two basic mechanisms by which these attacks work are firstly by exhausting resources, for example, by flooding the server with half-open connection requests, and secondly, by sending malformed packets, thereby making it unavailable to intended users. An elevated version of DoS attacks is termed as DDoS (distributed DoS) that can scale the attack many times by engaging large number of victim machines, termed as botnet, in the attack [48]. Disruption of service can be caused by three attack vectors mentioned in Figure 7 that fall under the category of DDoS attacks.

Many conventional supervised/semisupervised/unsupervised ML methods have been proposed and performance comparison done to detect distributed DoS attacks [49, 50]. Deep learning methods like convolutional neural network, LSTM, and recurrent neural networks are also applicable if traffic characteristics can be represented in a suitable way [51, 52]. Intrusion aiming at DoS was detected using ML approaches and demonstrated on UNSW-NB15 dataset [53]. Association rule mining, naïve Bayes, decision trees, and ANN classifiers have been used. Decision trees gave best accuracy of an order of about 94%. However, neither the proposed method is real time nor the metrics-based evaluation.

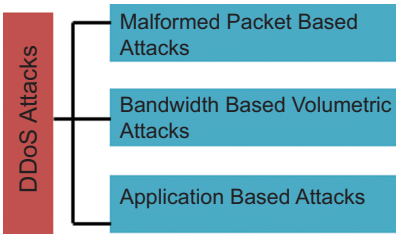


Figure 7: Classification of DDoS attacks.

Botnets: As mentioned earlier, DDoS attack involves the use of many compromised remote machines known as “bots” in targeting a server. Random forest and multilayer perceptron models have been used for botnet detection where analytics focus on

finding optimal window sizes so as to detect the botnet as soon as it starts compromising the system [54]. For complete security, merely checking ingress traffic is not sufficient. Egress traffic also needs to be monitored, for example, for malicious interaction like a host compromised as bot, trying to connect to its command-and-control server. Classification and regression tree algorithms and decision trees have been used for feature engineering and subsequent feature selection of bot detection.

4.5.1 Detection of DDoS attacks

Signature-based and anomalous behavior-based detection are two main methods of detecting known and unknown types of attacks, respectively. Signature-based methods suffer from the drawback of detecting zero-day attacks, and anomaly-based methods find them sometimes difficult to differentiate smart attack traffic like low-rate DDoS from normal traffic. A few IDS are now based on hybrid techniques also [55].

In order to differentiate normal traffic from DDoS attack, information statistics like entropy in transmitted data, total number of bytes exchanged between a pair of machines, number of unique pairs of source and destination IP addresses, and other field-wise statistics of both transmission control protocol and Internet protocol are considered [56].

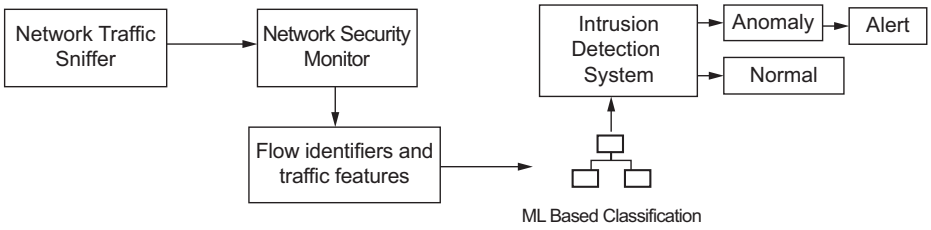


Figure 8: ML-based network behavior anomaly detection.

Figure 8 shows steps for applying ML to analyze network behavior for a possible DoS/DDoS attack. Normal and malicious behavior in network traffic can be differentiated by analyzing statistics of data from routers and switches. This can be used as a detection and response system by finding any deviations from expected patterns [57]. Whether the abnormal behavior is due to the new type of benign traffic or unknown attack can be detected using ML techniques. k-Means clustering along with network data mining can learn cluster centroids such that intervals of normal traffic are well separated from anomalous traffic. k-Medoids algorithm that performs better with noisy data has also been used. Expectation maximization clustering-based anomaly detection mechanisms have also been found to outperform other clustering methods. Hybrid approaches to improve efficiency have also been proposed.

For example, a hybrid k-means and ID3 decision tree has been utilized to find abnormal traffic in address resolution protocol exchanges. ML for intrusion detection has been shown to perform well on KDD CUP99 benchmark dataset. Experimental results have shown that ANN and SVM are effective techniques for anomaly detection. Once the model has been created, it can be applied to detect deviations from predefined threshold to detect anomalies.

5 Limitations of machine learning-based solutions in cybersecurity

Studies on ML-based successful analytics for cyberattack characterization may not be the complete picture. In the hindsight, several open problems have been cited by researchers. As most of the solutions are based on supervised ML, difficulty of obtaining ground truths will continue to haunt as and when these approaches would be applied to slightly different scenarios. Another important problem that has not been addressed sufficiently is about handling concept drift. The problem of concept drift occurs with passage of time and there is a shift in behavior/values of legitimate data [58]. Solutions like on-the-fly update of the predictive models with respect to new training data, model flexibility to change underlying classifiers whenever system performance falls below the threshold, and keeping the human in loop are some that have been proposed. However, there is a lack of detailed works which study the effectiveness of these proposals.

Present ML techniques are good at automating certain security monitoring tasks, but due to several pitfalls in their implementation, overall effectiveness is diminished. Thus, security analyst triaging of false alarms has not been reduced much. The following problems are inherent in ML-based models.

5.1 Data problems

Any ML system is as good as the data it gets.

a) Classifier performances are only as good as the relevance and quality of its training dataset. If updated training set covering all types of samples is not available, then none of the algorithms will perform well in long term.
b) On real traffic data, general-purpose classifiers are not very accurate as compared to specific-attack ML-based detectors.
c) Lack of standard benchmarking datasets: Datasets including a large number of real-world samples from diverse threat scenarios will be more useful for guaranteed performance [61].

d) Lab versus real traffic: Vendor-provided ML models are also trained on datasets of their laboratories and not real-world data.
e) Retraining issues: Controlled, repeatable, and efficient processes for capturing datasets from the application domain for updated learning at regular intervals are difficult in practical. Organizations do not consider such type of costs as relevant and thus assign low priority. As a result, for research also models cannot be retrained with newer, more relevant data.
f) Concept drift: After a period of running, ML systems may become irrelevant (also known as "model rot"), as their performance may deteriorate with time. This may happen due to the evolving input data owing to some external effects.

5.2 Model problems

Problems due to ML solutions are provided by the vendor:

a) Vulnerability to adversarial attacks: Adversaries are using novel strategies to evade detectors based on ML algorithms [59]. They may target to attack integrity, availability, and privacy of ML-based system. Integrity violations lead to misclassifications. Availability violations may make the model look rogue by raising false alarms at a high rate. Privacy attacks are about acquiring information not supposed to be public. Recently developed GANs are shown to be useful in attacking ML-based autonomous system with poisoned data points while remaining under the radar. A detailed study of this for autonomous cars has been presented in [60].
b) Lack of new features design: Very few studies focus on engineering novel features demonstrating high quality of distinguishability for various types of threats. Moreover, in most cases, new features have to be hand-engineered.
c) Lack of new classifier design: ML algorithms are general purpose, if tailored to the characteristics threat/attack analysis and discovery.
d) Model quality: Benchmarking security solutions by testing with same settings are not available, and hyperparameter optimization is still an issue.
e) Problems like polymorphism have been studied from malware point of view but not for other security applications like spam detection, phishing detection, and intrusion detection.
f) Model extraction attack: With access only to test samples and results from the black-box classifier, labeled training dataset can be generated with which we can train a local substitute model. This local substitute model can be analyzed offline to search for samples that belong to adversarial space.
g) Existing ML approaches resistant to data quality problems like bias in data and label inaccuracy.
h) Warm start problems: ML models may have to be updated on the arrival of significant new data. If the new prediction model is trained using the existing model as a starting point (this process is called "warm start"), then it may perform worse.

i) Lack of an integrated model for holistic cybersecurity solutions. Currently, there are multiple models for multiple problems.
j) Scaling: One of the biggest challenges around using ML is that solutions are proposed on toy problems and getting it working in large live networks is not very smooth.
k) None of the ML solutions guarantee threshold false positives (FPs) and true negatives (TNs). A high number of FPs are annoying and false negatives lead to compromise. Moreover, each FP and TN incurs the cost of follow up too.
l) Lack of studies on integration across models and domains for creating collective intelligence.
m) Lack of semantics: Semantic gap is a problem with ML. Compared to static rulesets or heuristics, it is not directly explainable as to why an event was flagged as an anomaly. This can lead to reduced confidence in predictions. A human-readable explanation for alerts generated by ML systems is a practical requirement that is missing in current systems.
n) Increased time to detection in stealthy attacks that hide between normal traffic is yet to be addressed sufficiently.
o) Scope of connecting cross-domain information: For example, if there is an increase in the number of DGA domains being detected, this input must be provided to malware detectors. Reduced attacks from some specific adversary also need attention to find out if it is attacking from some other unknown front or has worked out good detection evasion method. There is very little research exploring these aspects in cybersecurity domain.

6 Opportunities for new ML paradigms

Adversarial training: Recent advances in adversarial ML have proved that robust ML may not be a reality. A well-performing model may misclassify frequently when poisoned with adversarial examples. To counter adversarial attacks, proactive adversarial training can be used [62]. It involves augmenting the training dataset of supervised models with adversarial examples and adding their correct labels. Methods used to poison models can be regenerated in a similar way and preadded to the training models.

Reinforcement learning: In this paradigm of ML, labeled data are not available. Thus, the algorithms learn to make predictions on their own. There are agents who know the start and end states of goal but not the path. A reward-based path finding will every time the agent finds solutions in the most optimal way. In this manner, the agent learns that own. This branch of ML would be very helpful in applying cybersecurity in areas where labeled dataset is not available. For example, processing CCTV camera feeds for real-time detection of suspicious activities can be done by

reinforcement learning. Recently, reinforcement learning has been applied for intrusion detection, IoT security, network penetration testing, and malware analysis [63, 64].

Active learning: Active learning, as the name suggests, is a type of ML where the learner chooses the dataset for training. In this way, if the learner optimally chooses the dataset, then the desired accuracy can be achieved by a much smaller training size as compared to traditional ML that tries to ingest all which is available. An active learner may find important data instances and ask queries to obtain their labels [65]. This type of ML can find a very good application in cybersecurity where most of the datasets are imbalanced and very few attack instances are labeled.

Distributed machine learning (DML): As mentioned earlier, network logs, malware, and all other security data emanate at a huge pace. There is a dearth of processing power at one place or in one organization to process all these data or train the ML models in a centralized way with all the available data . A new concept of distributed ML that facilitates distributing the ML workload across multiple machines would be really useful in this case [66]. Once the challenges with such distributed systems like efficient parallelization of the training process and integrating models are addressed, there will be huge opportunities for implementation of DML in cybersecurity .

Federated machine learning: This is another paradigm to scale up ML-based learning systems. While in DML, data available at one location can be distributed to multiple computing power facilities to have parallel learning, in federated learning, datasets at various sites are used to learn small in-place models, and then the learnt models are federated to obtain a robust integrated ML model based on huge variety of training datasets [67]. Organizations can enter into agreements to share models developed on their own datasets and contribute to a federated more robust learner. Till now, the use of federated learning has been very limited in the cybersecurity domain, although it has huge promises in this field as organizations are wary to share their raw data but may be willing to share models once secure federated learning becomes realizable.

7 Conclusions

In this work, applications of ML techniques in various cybersecurity domains for attack prevention, detection, and response have been studied. It has been observed from the literature that although there are certain pitfalls like data-related problems of unbalanced datasets, statistical heterogeneity, and concept drift, the trend to incorporate ML capabilities into new and existing security products will continue its pace, thus enhancing the reliability of automated cybersecurity.

References

[1] Craigen, D., Diakun-Thibault, N., & Purse, R. (2014). Defining cybersecurity. *Technology Innovation Management Review*, *4*(10).

[2] Jang-Jaccard, J., & Nepal, S. (2014). A survey of emerging threats in cybersecurity, *Journal of Computer and System Sciences*, *80*(5), 973–993.

[3] Saravanan, A., & Bama, S. S. (2019). A review on cyber security and the fifth generation cyberattacks. *Oriental Journal of Computer Science and Technology*, *12*(2), 50–56.

[4] Landauer, M., Skopik, F., Wurzenberger, M., & Rauber, A. (2020). System log clustering approaches for cyber security applications: A survey. *Computers & Security*, *92*, 101739.

[5] Sarker, I. H., Kayes, A. S., Badsha, S., Alqahtani, H., Watters, P., & Ng, A. (2020). Cybersecurity data science: An overview from machine learning perspective. *Journal of Big Data*, *7*(1), 1–29.

[6] https://www.gartner.com/smarterwithgartner/gartner-top-security-and-risk-trends-for-2021 [Accessed on 27-11-2021]

[7] Onwubiko, C. (2017 Jun 19). Security operations centre: Situation awareness, threat intelligence and cybercrime. In 2017 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA) (pp. 1–6). IEEE.

[8] Pranggono, B., & Arabo, A. (2021). COVID-19 pandemic cybersecurity issues. *Internet Technology Letters*, *4*(2), e247.

[9] https://us.norton.com/internetsecurity-emerging-threats-cyberthreat-trends-cybersecurity-threat-review.html [Accessed on 27-11-2021]

[10] Chapman, I. M., Leblanc, S. P., & Partington, A. (2011 Apr 3). Taxonomy of cyber attacks and simulation of their effects. In Proceedings of the 2011 Military Modeling & Simulation Symposium (pp. 73–80).

[11] Hussain, A., Mohamed, A., & Razali, S. (2020 Mar 31). A Review on Cybersecurity: Challenges & Emerging Threats. In Proceedings of the 3rd International Conference on Networking, Information Systems & Security (pp. 1–7).

[12] Nam, T. (2019). Understanding the gap between perceived threats to and preparedness for cybersecurity. *Technology in Society*, *58*, 101122.

[13] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Machine learning basics. *Deep Learning*, *1*(7), 98–164.

[14] Dasgupta, D., Akhtar, Z., & Sen, S. (2020 Sep). Machine learning in cybersecurity: A comprehensive survey. *The Journal of Defense Modeling and Simulation*, 1548512920951275.

[15] Salloum, S. A., Alshurideh, M., Elnagar, A., & Shaalan, K. (2020. Machine learning and deep learning techniques for cybersecurity: A review. In *AICV* (pp. 50–57).

[16] Shaukat, K., Luo, S., Varadharajan, V., Hameed, I. A., Chen, S., Liu, D., & Li, J. (2020 Jan). Performance comparison and current challenges of using machine learning techniques in cybersecurity. *Energies*, *13*(10), 2509.

[17] Sun, N., Zhang, J., Rimba, P., Gao, S., Zhang, L. Y., & Xiang, Y. (2018Dec7). Data-driven cybersecurity incident prediction: A survey, *IEEE Communications Surveys & Tutorials*, *21*(2), 1744–1772.

[18] Li, J., Zhao, B., & Zhang, C. (2018 Dec). Fuzzing: A survey. *Cybersecurity*, *1*(1), 1–3.

[19] Babiker, M., Karaarslan, E., & Hoscan, Y. (2018 Mar 22). Web application attack detection and forensics: A survey. In 2018 6th IEEE international symposium on digital forensic and security (ISDFS) (pp. 1–6).

[20] Jain, A. K., & Gupta, B. B. (2018). Towards detection of phishing websites on client-side using machine learning based approach. *Telecommunication Systems*, *68*(4), 687–700.

[21] Sahoo, D., Liu, C., & Hoi, S. C. (2017 Jan 25). Malicious URL detection using machine learning: A survey. arXiv preprint arXiv:1701.07179.
[22] Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019 Mar). Machine learning based phishing detection from URLs. *Expert Systems with Applications*, *1*(117), 345–357.
[23] Gangavarapu, T., Jaidhar, C. D., & Chanduka, B. (2020). Applicability of machine learning in spam and phishing email filtering: Review and approaches. *Artificial Intelligence Review*, *53*(7).
[24] Annadatha, A., & Stamp, M. (2018 Feb). Image spam analysis and detection. *Journal of Computer Virology and Hacking Techniques*, *14*(1), 39–52.
[25] Latchoumi, T. P., Reddy, M. S., & Balamurugan, K. (2020). Applied machine learning predictive analytics to SQL injection attack detection and prevention. *European Journal of Molecular & Clinical Medicine*, *7*(02), 2020.
[26] https://github.com/gyoisamurai/GyoiThon [Accessed on 29 Nov, 2021]
[27] Le, D. C., Zincir-Heywood, N., & Heywood, M. I. (2020Jan17). Analyzing data granularity levels for insider threat detection using machine learning, *IEEE Transactions on Network and Service Management*, *17*(1), 30–44.
[28] https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099 [Accessed on 29 Nov, 2021]
[29] Le, D. C., & Zincir-Heywood, A. N. (2019 Apr 8). Machine learning based insider threat modelling and detection. In 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM) (pp. 1–6). IEEE.
[30] https://www.ftc.gov/reports/consumer-sentinel-network-data-book-2020 [Accessed on 29 Nov, 2021]
[31] https://en.wikipedia.org/wiki/2017_Equifax_data_breach [Accessed on 29 Nov, 2021]
[32] Galbally, J., Coisel, I., & Sanchez, I. (2017). A new multimodal approach for password strength estimation – Part II: Experimental evaluation, *IEEE Transactions on Information Forensics and Security*, *12*(12), 2845–2860.
[33] Hitaj, B., Gasti, P., Ateniese, G., & Perez-Cruz, F. (2019 Jun 5). Passgan: A deep learning approach for password guessing. In International Conference on Applied Cryptography and Network Security (pp. 217–237). Springer, Cham.
[34] Takaaki, S., & Atsuo, I. (2019 Mar 13). Dark web content analysis and visualization. In Proceedings of the ACM International Workshop on Security and Privacy Analytics (pp. 53–59).
[35] Yang, M., Gu, X., Ling, Z., Yin, C., & Luo, J. (2017). An active de-anonymizing attack against tor web traffic, *Tsinghua Science and Technology*, *22*(6), 702–713.
[36] Alshamrani, A., Myneni, S., Chowdhary, A., & Huang, D. (2019). A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities, *IEEE Communications Surveys & Tutorials*, *21*(2), 1851–1877.
[37] Singh, S., Inamdar, A., Kore, A., & Pawar, A. (2020 Jul 28) Analysis of Algorithms for User Authentication using Keystroke Dynamics. In 2020 International Conference on Communication and Signal Processing (ICCSP) (pp. 0337–0341). IEEE.
[38] Vinayakumar, R., Soman, K. P., Velan, K. S., & Ganorkar, S. (2017 Sep 13). Evaluating shallow and deep networks for ransomware detection and classification. In 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 259–265). IEEE.
[39] Ghaffarian, S. M., & Shahriari, H. R. (2017). Software vulnerability analysis and discovery using machine-learning and data-mining techniques: A survey, *ACM Computing Surveys (CSUR)*, *50*(4), 1–36.

[40] She, D., Pei, K., Epstein, D., Yang, J., Ray, B., & Jana, S. (2019 May 19). Neuzz: Efficient fuzzing with neural program smoothing. In 2019 IEEE Symposium on Security and Privacy (SP) (pp. 803–817). IEEE.
[41] Maksutov, A. A., Morozov, V. O., Lavrenov, A. A., & Smirnov, A. S. (2020 Jan 27). Methods of deepfake detection based on machine learning. In 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus) (pp. 408–411). IEEE.
[42] Khan, J. Y., Khondaker, M., Islam, T., Iqbal, A., & Afroz, S. A benchmark study on machine learning methods for fake news detection. arXiv preprint arXiv:1905.04749. 2019 May.
[43] Ahmed, H., Traore, I., & Saad, S. (2017 Oct 25). Detection of online fake news using n-gram analysis and machine learning techniques. In International conference on intelligent, secure, and dependable systems in distributed and cloud environments (pp. 127–138). Springer, Cham.
[44] Ucci, D., Aniello, L., & Baldoni, R. (2019). Survey of machine learning techniques for malware analysis. *Computers & Security*, *1*(81), 123–147.
[45] Singh, J., & Singh, J. (2021). A survey on machine learning-based malware detection in executable files. *Journal of Systems Architecture*, *112*, 101861.
[46] Li, J., Sun, L., Yan, Q., Li, Z., Srisa-An, W., & Ye, H. (2018). Significant permission identification for machine-learning-based android malware detection, *IEEE Transactions on Industrial Informatics*, *14*(7), 3216–3225.
[47] Alqurashi, S., & Batarfi, O. (2016). A comparison of malware detection techniques based on hidden Markov model, *Journal of Information Security*, *7*(3), 215–223.
[48] Douligeris, C., & Mitrokotsa, A. (2004). DDoS attacks and defense mechanisms: Classification and state-of-the-art, *Computer Networks*, *44*(5), 643–666.
[49] Lima Filho, F. S., Silveira, F. A., de Medeiros Brito Junior, A., Vargas-Solar, G., & Silveira, L. F. (2019). Smart detection: An online approach for DoS/DDoS attack detection using machine learning. *Security and Communication Networks*, *2019*.
[50] Tuan, T. A., Long, H. V., Son, L. H., Kumar, R., Priyadarshini, I., & Son, N. T. (2020). Performance evaluation of Botnet DDoS attack detection using machine learning. *Evolutionary Intelligence*, *13*(2), 283–294.
[51] Haider, S., Akhunzada, A., Mustafa, I., Patel, T. B., Fernandez, A., Choo, K. K., & Iqbal, J. (2020). A deep CNN ensemble framework for efficient DDoS attack detection in software defined networks. *IEEE Access*, *8*, 53972–53983.
[52] Li, Q., Meng, L., Zhang, Y., & Yan, J., Zhai, G., Zhou, J., An, P., Yang, X. (2018). DDoS attacks detection using machine learning algorithms. In *International forum on digital TV and wireless multimedia communications* (pp. 205–216). Springer, Singapore.
[53] Meftah, S., Rachidi, T., & Assem, N. (2019). Network based intrusion detection using the UNSW-NB15 dataset. *International Journal of Computing and Digital Systems*, *8*(5), 478–487.
[54] Santana, D., Suthaharan, S., & Mohanty, S. (2018 May 3). What we learn from learning-Understanding capabilities and limitations of machine learning in botnet attacks. arXiv preprint arXiv:1805.01333.
[55] Cepheli, Ö., Büyükçorak, S., & Karabulut Kurt, G. (2016). Hybrid intrusion detection system for DDoS attacks. *Journal of Electrical and Computer Engineering*, *2016*.
[56] Aamir, M., & Zaidi, S. M. (2019). DDoS attack detection with feature engineering and machine learning: The framework and performance evaluation. *International Journal of Information Security*, *18*(6), 761–785.
[57] Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, *60*, 19–31.
[58] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2018). Learning under concept drift: A review, *IEEE Transactions on Knowledge and Data Engineering*, *31*(12), 2346–2363.

[59] Ibitoye, O., Abou-Khamis, R., Matrawy, A., & Shafiq, M. O. (2019 Nov 6). The threat of adversarial attacks on machine learning in network security – a survey. arXiv preprint arXiv:1911.02621.
[60] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018). Generative adversarial networks: An overview, *IEEE Signal Processing Magazine*, *35*(1), 53–65.
[61] Yavanoglu, O., & Aydos, M. (2017). A review on cyber security datasets for machine learning algorithms. In 2017 IEEE international conference on big data (big data) (pp. 2186–2193). IEEE.
[62] Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., & McDaniel, P. (2017 May 19). Ensemble adversarial training: Attacks and defenses. arXiv preprint arXiv:1705.07204.
[63] Uprety, A., & Rawat, D. B. (2020). Reinforcement Learning for IoT security: A comprehensive survey. *IEEE Internet of Things Journal*.
[64] Dang, Q. V., & Vo, T. H. (2022). Reinforcement learning for the problem of detecting intrusion in a computer system. In *Proceedings of sixth international congress on information and communication technology* (pp. 755–762). Springer, Singapore.
[65] Rhee, P. K., Erdenee, E., Kyun, S. D., Ahmed, M. U., & Jin, S. (2017). Active and semi-supervised learning for object detection with imperfect data. *Cognitive Systems Research*, *45*, 109–123.
[66] Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., & Rellermeyer, J. S. (2020). A survey on distributed machine learning, *ACM Computing Surveys (CSUR)*, *53*(2), 1–33.
[67] Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications, *ACM Transactions on Intelligent Systems and Technology (TIST)*, *10*(2), 1–9.

Vasu Thakur, Vikas Kumar Roy*, Nikhil Baliyan, Nupur Goyal, Rahul Nijhawan

A framework for seborrheic keratosis skin disease identification using Vision Transformer

Abstract: Over the past decades, skin diseases have been a hazardous issue because of more sophisticated and high-cost treatments. A noncancerous disease, seborrheic keratosis, treatment in the initial stages is very important to avoid the spread on the skin over time. Identifying the skin disease is still a stimulating mission for dermatologists. This chapter proposes an effective approach by using Vision Transformers (ViT) to identify seborrheic keratosis, and obtains 99% accuracy. We considered the data that is publicly available on Kaggle, which comprises 386 images. We did the comprehensive observation to get better results using ViT that applies self-attention with transformer architecture to the image patch sequence and compares with different convolutional neural network (CNN) algorithms (VGG-19 and Inception-V3) with ViT to analyze the distinction between the proposed approach and other CNN models. A comparative study for the classification in CNN models has also been provided.

Keywords: Vision Transformer, Inception-V3, VGG-19, skin cancer

1 Introduction

Skin is the largest organ and is the exterior surface of the human body, which serves as a protective shield against heat, sunlight, injury, and infection, and also regulates the body temperature. There is a very high probability of getting infected with a polluted environment every time the body comes in contact with it. People have been avoiding skin-related diseases due to ignorance and lack of medical knowledge and have casual concern toward it. Furthermore, many diseases under our skin cannot be diagnosed in the early stages unless these are mutated, spread,

***Corresponding author: Vikas Kumar Roy,** Department of Computer Science and Engineering, Roorkee Institute of Technology, Roorkee, India, e-mail: royvikas1610@gmail.com
Vasu Thakur, Nikhil Baliyan, Department of Computer Science and Engineering, Roorkee Institute of Technology, Roorkee, India
Nupur Goyal, Department of Mathematics, Graphic Era Deemed to be University, Dehradun, India, e-mail: nupurgoyalgeu@gmail.com
Rahul Nijhawan, Department of Computer Science and Engineering, University of Petroleum and Energy Studies, Dehradun, India, e-mail: rahul.nijhawan@ddn.upes.ac.in

https://doi.org/10.1515/9783110766745-006

and infect the other body parts, and this growth causes the cosmetic problem, and we do not like how they look [1]. At this specific stage, disease diagnosing is very complex, and having control over it becomes impossible. Various skin diseases tend to occur within families also, and genes may be the cause for these diseases.

Seborrheic keratoses are harmless skin growths that bear a resemblance to skin cancer and often increase the growth with time. It has the tendency to develop from one to several, in general. These diseases appeared as brown and patchy and can be random on the skin. In an initial phase, it is mistaken as unusual-looking scabs by many people. Seborrheic keratoses do not typically cause symptoms, and these growths cause cosmetic problems [1].

In the past few decades, many researchers are focused on identifying the skin disease at the initial stages to detect the seborrheic keratoses. A comparative analysis of the skin disease with deep learning techniques is done, such as VGG-19, Inception-V3, and Vision Transformer (ViT). It is not clear what exactly causes the seborrheic keratosis and is important to detect at the initial stages because it is common to develop several from one.

We have proposed a framework for skin disease identification using ViT that uses a group of CNN and it has outperformed the state-of-the-art deep learning. We have given a detailed comparison with various CNN algorithms. Statistical analysis and comparison of the different state-of-the-art models like area under the curve (AUC), precision, and accuracy are done with the proposed approach and obtained the best accuracy than others.

ViT is an upgraded variant of transformer which requires linear information to perform accurately. This model splits the images into small patches, which are considered sequence tokens, and afterward, patches flattening takes place. It generates low-dimensional embeddings linearly with the flattened patches to keep relevant information, and finally, the sequence output is passed to the transformer encoder as an input. Without the usage of convolution layers, it transforms the image into small-sized patches to avoid the difficulties of other algorithms.

Rest of the chapter is structured as follows: Section 2 discusses the similar type of work conducted in the literature, Section 3 discusses the dataset, and Section 4 discusses the methodology adopted. Section 5 elaborates on the working or implementation of CNN. Section 6 explains the implementation of proposed approach. Section 7 contains the results and discussion, and finally, Section 8 concludes with future directions.

2 Literature review

The evolution of convolutional neural networks (CNNs) has led to the great focus on disease detection at very early stages. An advanced method of machine learning

is deep learning, which is a neural network with more than two layers that works like a human brain [2]. Indi and Gunge [3] explained a deep learning methodology as a neural network learning process, and one of the important features of deep learning is that it can automatically acquire the image features from image patterns. Kang and Chen [4] proposed a framework that contains a fruit detector based on deep learning called "LedNet" and observed the apple detection in orchards effectively. Kaizhou Li et al. [5] applied the architecture of CNN of Inception-V3 models to categorize the various notches of ginkgo leaf disease and achieved the accuracy of 92.3% under the fixed laboratory environments and 93.2% under field environments. Celebi [6] proposed a methodological way to identify the skin lesions on dermoscopic images by separating the scratch from the contextual skin. The instability of the classifier is evaluated using Monte Carlo cross-validation. In [6], the author used a set of 564 images, and the result accuracy is 93.3%. Sarode et al. [7] calculated the nutrition and calories present inside the food with an image processing system that takes food images as input and extracts the features using CNN. Jia et al. [8] proposed a mask region-CNN to identify the overlapped apples and determined the accuracy of 97.31% as the rate of identifying apple image became swifter. Esteva et al. [15] discussed the classification of skin lesions using a single CNN and trained an end-to-end model from images directly using only pixels and disease markers as inputs. Yujian and coworkers [16] focused on fine-tuning and estimated the advanced deep CNN for image-based plant disease cataloguing. Liu et al. [17] proposed a deep learning model that provides a better solution to control the apple leaf diseases with high precision and a high convergence rate. The image generation technique has also been proposed by Liu to improve the strength of the CNN model [17].

Our proposed approach ViT achieved the top-5 accuracy of 99%. Following various architectures of CNN models are VGG-19 and achieved an accuracy of 95.56%, and InceptionV3 with an accuracy of 95.19% with a neural network. The architecture of ViT is analyzed and used on patches of images in the dataset. Therefore, we proposed the ViT for detecting seborrheic keratoses.

3 Data collection

We have acquired a dataset from Kaggle. The dataset consists of 386 normal and seborrheic keratoses images. We have only considered the images of good quality to improve the accuracy of our proposed approach as revealed in Figure 1.

The dataset is divided into two parts: training data comprises 266 images and testing data contains 120 images. The total images available in our dataset are detailed in Table 1.

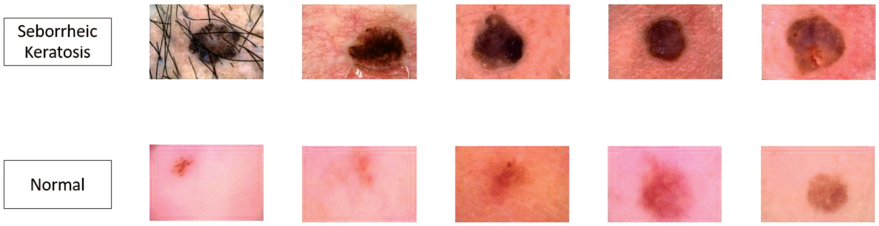


Figure 1: Sample images of skin from our dataset.

Table 1: Number of images in the dataset.

Skin	Number of images
Seborrheic keratoses	386
Normal	386

4 Methodology

Our methodology consists of three phases such as acquisition of dataset, feature extraction, and cataloguing, as given in Figure 2. Firstly, the images which belong to seborrheic keratoses are passed as input and split into patches, and afterward, the output is sequenced linearly. Furthermore, the model is trained by the tokens (which are the image patches) in an organized way. We have made a comparison among support vector machine (SVM), arbitrary forest, K-nearest neighbors (KNN), naïve Bayes, and so on, with the proposed approach to get better results.

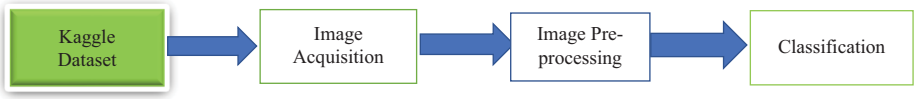


Figure 2: Seborrheic keratosis disease recognition methodology.

5 Convolutional neural network (CNN)

Contemporary CNN is an advanced technique due to its ability of feature extraction in pictures without multifaceted preprocessing [18], attached with transfer erudition and fine-tuning constraints. It uses various kernels to process images as input [19]. CNNs are being used in various aspects because of their working efficiencies and

the expected accuracy obtained such as image classification and speech recognition [9]. The output of CNN models acts as an input of a KNN model that categorizes the seborrheic keratoses of the affected skin with normal skin.

5.1 VGG-19

VGG-19 architecture comprises 16 convolution layers, 3 fully connected layers, 5 max pool layers, and one SoftMax layer. The 224 × 224 RGB images act as an input, where 3 × 3 kernel size is used. It allows the dimensional stuffing to build the image resolution, and subsequently, rectified linear unit is accessible for the improvement of less computation. It uses various convolutional and nonlinear activation layers [10].

5.2 Inception v3

The architecture is a CNN from the Inception domain making assorted furtherance including label smoothing, factorized 7 × 7 convolution [11], and the use of auxiliary classifiers to disseminate label particular, as sliding down the networks.

This CNN architecture is used to comprehend the important structures that required to classify seborrheic keratoses disease.

6 Vision Transformer

Over the past, CNNs have been used to solve most computer vision problems [12] and overcome its limitation. These CNNs have been performed successfully. Because of the linear functions of recurrent models, they cannot assist parallelization, which affects the long-term learning of the model. ResNet resolves the sophisticated identification of images on the ImageNet [20].

In CNN, the image is made up of pixels, reliant on the other neighboring pixels. For feature extraction, it uses strainers on image patches [21]. CNN trains the model on the selected features rather than considering all the features to make the model more precise. The drawback of CNN is that it does not care about the feature's relative situation and cannot encode the spatial instruction. On the other hand, ViT trains the model, such as relative feature position, encoding spatial instruction, and performing well than different advanced algorithms.

We proposed the ViT approach to identify seborrheic keratoses. In the past few years, the ViT [13] has opted for large datasets, and we applied a self-attention mechanism to the patches of images to predict accurately.

The first deep neural network algorithm was introduced as ViT. This is used for large-scale computer visualization datasets without any convolution machinists. It employs the novel transformer established in NLP problems with limited variations [14]. Firstly, it divides the image into small-sized image patches. Tokens produced in the NLP application are the same as these image patches. Secondly, flattening the image patches is done and then resized them into the vector format, which is passed as an input to the transformer. After the vectorization of images, it converts into the embeddings linearly. Simultaneously, position embedding is passed through each of the embedding patches to conserve the information of position of every entrenching area of the network.

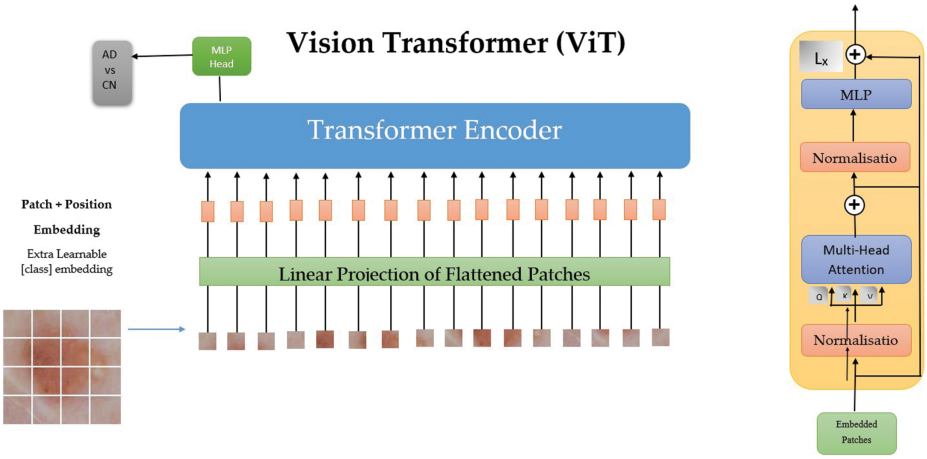


Figure 3: ViT model architecture.

Although architecture of ViT is just like a language transformer. The output provided by the transformer encoder is nourished directly to an multilayer perceptron (MLP) head where there is no decoder support. The image of seborrheic keratoses disease is converted into a vector by 2D flattening which is the input prerequisite of the transformer. The input image is divided into equal-sized patches and implanted with rectilinear prognosis. Position entrenching is added, and the resulting arrangement is nourished to a transformer's encoder. Like BERT's method, in ViT also, a class token is accompanying to the commencement of the arrangement of entrenched patches as a learnable entrenching.

The transformer encoder is the preeminent architecture comprising numerous encoder blocks, and every block has multiple self-attention mechanisms. The normalization of each layer is passed as an input to the multihead self-attention (MSA) tool and MLP blocks [12]. In Figure 3, the basic structure of an encoder has been exposed, which explained the working principle of an encoder's instrument, multihead consideration, and scaled-dot product attention.

7 Results and discussion

Here, we discuss the findings obtained by the ViT architecture – the proposed approach. The accuracy of our framework is the best at top-5 accuracy of 99.4% compared to another CNN architecture. To obtain the results, different datasets are used preliminarily on trained models. Here, we discuss three scenarios.

In the first situation, authors proposed a more efficient technique by using a ViT that follows the self-engrossment mechanism, which is an efficient method to update the parallel input text to the embedding. We divide the original image into patches of small in size, which then passed to the linear embedding sequentially as an input to transform it into an n-dimensional vectorized image, and attained a noteworthy result of accuracy of 99.4%.

In the second scenario, we proposed a VGG-19 architecture that comprises 16 convolution films, 3 fully connected films, 5 max pool films, and 1 softmax film and 224 × 224 RGB images that act as an input where 3 × 3 kernel size is used. It allows the dimensional stuffing to build the image resolution, and subsequently, rectified linear unit is accessible for the improvement of less computation [11]. Furthermore, the accuracy achieved by VGG-19 is shown in Table 2. Using this architecture, an accuracy of 95.56% is attained with the neural network, 95.194% with stack, 94.82% with logistic regression, 93.90% with KNN, 92.23% with random forest, 91.866% with SVM, and 85.95%.

In scenario 3, the Inception-V3 model works completely on convolution that splits $n \times n$ convolution kernel into $1 \times n$ and $n \times 1$. These are responsible for decreasing the number of parameters in training [11]. The main drawback of Inception-V3 is the absence of the residual association in the network to accomplish more precision in image classification. We achieved an accuracy of 95.194% with the neural network, 94.63% with stack, 94.63% with logistic regression, and 94.45% with KNN, 91.68% with random forest, 95.00% with SVM, and 90.00% with naive Bayes using this architecture.

It is observed that the highest accuracy has been achieved with ViT. We get the top-5 accuracy of 99.4% on our test data and the second highest accuracy achieved by VGG-19 with a neural network classifier.

The AUC, accuracy, and precision of VGG-19, Inception-V3, and ViT are shown in Table 2.

The precision calculation of feature extraction of ViT and various CNN models (VGG-19 and Inception-V3) are shown and found that the ViT has the uppermost precision of 99.4% when compared with CNN architecture, which is better.

Binary cataloguing outcomes of seborrheic keratoses disease are revealed in Figure 4, which represent the achieved accuracy of VGG-19, ViT, and Inception-V3. The SVM, KNN, and various other ML algorithms are used as classifier, and their obtained accuracy is shown. *Y*-axis shows the accuracy obtained in all three scenarios and *X*-axis represents the corresponding models.

Table 2: Comparative Study.

Models	Feature extractor	AUC	Accuracy	Precision
VGG-19 [11]	K-nearest neighbors (KNN)	0.97	0.94	0.94
	Tree	0.86	0.89	0.89
	Stack	0.98	0.95	0.95
	SVM	0.98	0.92	0.92
	Random forest (Rf)	0.97	0.92	0.92
	Neural network	0.98	0.95	0.95
	Naive Bayes	0.88	0.86	0.86
	Logistic regression	0.99	0.95	0.95
	CN2 rule inducer	0.83	0.75	0.75
Inception-V3 [10]	K-nearest neighbors (KNN)	0.98	0.94	0.94
	Tree	0.83	0.88	0.88
	Stack	0.99	0.95	0.95
	SVM	0.99	0.95	0.95
	Random forest (Rf)	0.96	0.91	0.91
	Neural network	0.99	0.95	0.95
	Naive Bayes	0.94	0.90	0.90
	Logistic regression (LR)	0.99	0.95	0.95
	CN2 rule inducer	0.83	0.73	0.73
Proposed approach	ViT	0.99	0.99	0.98

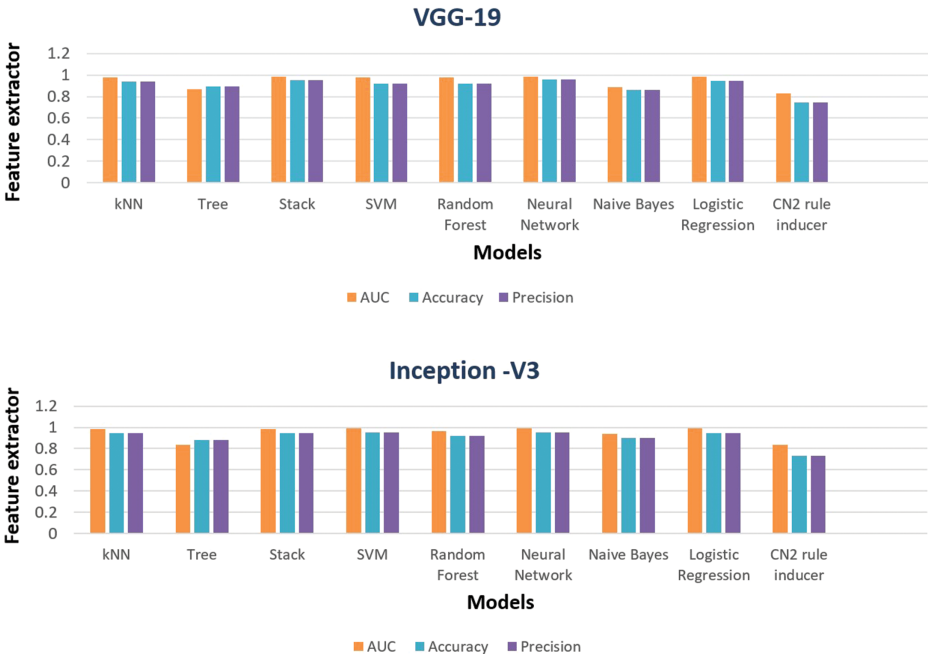


Figure 4: Binary classification results of seborrheic keratoses disease.

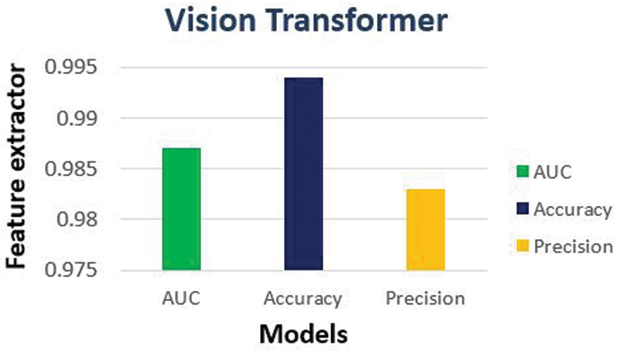


Figure 4 (continued)

7.1 Why Vision Transformer?

We have achieved the top-5 accuracy of 99%, which is better than the other described state-of-the-art algorithms. ViT uses a self-attention mechanism that allows the integration of the information entirely. This model computes in less time in training. It does not comprise convolution layer, but it can handle the variable-sized information that makes it highly preferable. Implementing the self-attention technique on the image patches is a sophisticated task without CNN. ViT resolves this drawback by dividing the image into patches of size 16 × 16. The transformer encode consists of MLP and MSA. The MSA divides the input into various heads to make the learning of each head easily. The result is associated and passed with the MLP. CNN functions on methods where pixels constitute the image in which every pixel is dependent on the neighboring pixels. ViT allows the positional embeddings of the image while CNN only fetches the relevant information and features for training. CNN does not encode the relative spatial instruction.

8 Conclusion

This chapter determined the accurate prediction of the skin disease "seborrheic keratoses" and compared our proposed approach with the other CNN models. This study concluded that our proposed method provides the results with upmost precision than VGG-19 and Inception-V3. This chapter is very effective for patients for disease detection in the early stages. In addition to the model, to improve cataloguing precision, data proliferation has also been accustomed.

Furthermore, the results obtained by this study can be very cooperative for the exposure of seborrheic keratoses in early stage and to take preventives to stop

further spread of the disease. Our work can be useful to the dermatologist and medical experts to identify the disease with limited human resources.

Acronyms

MLP Multilayer perceptron
MSA Multihead self-attention
CNN Convolutional neural network
ViT Vision Transformers

References

[1] Hafner, C., & Vogt, T. (2008). Seborrheic keratosis. *JDDG: Journal der Deutschen Dermatologischen Gesellschaft*, *6*(8), 664–677.
[2] Benuwa, B. B., Zhan, Y. Z., Ghansah, B., Wornyo, D. K., & Banaseka Kataka, F. (2016). A review of deep machine learning. *International Journal of Engineering Research in Africa*, *24*, 124–136.
[3] Indi, T. S., & Gunge, Y. A. (2016). Early stage disease diagnosis system using human nail image processing. *International Journal of Information Technology and Computer Science*, *7*, 30–35.
[4] Kang, H., & Chen, C. (2020). Fast implementation of real-time fruit detection in apple orchards using deep learning. *Computers and Electronics in Agriculture*, *168*, 105108.
[5] Li, K., Lin, J., Liu, J., & Zhao, Y. (2020). Using deep learning for image-based different degrees of ginkgo leaf disease classification. *Information*, *11*(2), 95.
[6] Garnavi, R., Aldeen, M., Celebi, M. E., Bhuiyan, A., Dolianitis, C., & Varigos, G. (2010). Automatic segmentation of dermoscopy images using histogram thresholding on optimal color channels. *International Journal of Medicine and Medical Sciences*, *1*(2), 126–134.
[7] Sarode, G., et al. Nutrition & calories within the food measured using processing & CNN method.
[8] Jia, W., et al. (2020). Detection and segmentation of overlapped fruits based on optimized mask R-CNN application in apple harvesting robot. *Computers and Electronics in Agriculture*, *172*, 105380.
[9] Indolia, S., Goswami, A. K., Mishra, S. P., & Asopa, P. (2018). Conceptual understanding of convolutional neural network-a deep learning approach. *Procedia Computer Science*, *132*, 679–688.
[10] Xiao, J., Wang, J., Cao, S., & Li, B. (2020, April). Application of a novel and improved VGG-19 network in the detection of workers wearing masks. *Journal of Physics. Conference Series*, *1518*(1), 012041. IOP Publishing.
[11] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2818–2826).
[12] Rezvantalab, A., Mitha, S., & Khademi, A. (2021). Alzheimer's Disease Classification using Vision Transformers.
[13] Chhabra, H. S., Srivastava, A. K., & Nijhawan, R. (2020). A hybrid deep learning approach for automatic fish classification. In *Proceedings of ICETIT 2019* (pp. 427–436). Springer, Cham.

[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

[15] Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, *542*(7639), 115–118.

[16] Too, E. C., Yujian, L., Njuki, S., & Yingchun, L. (2019). A comparative study of fine-tuning deep learning models for plant disease identification. *Computers and Electronics in Agriculture*, *161*, 272–279.

[17] Liu, B., Zhang, Y., He, D., & Li, Y. (2018). Identification of apple leaf diseases based on deep convolutional neural networks. *Symmetry*, *10*(1), 11.

[18] Kausar, A., et al. (2018). Pure-CNN: A framework for fruit images classification. In 2018 International Conference on Computational Science and Computational Intelligence (CSCI). IEEE.

[19] Momeny, M., et al. (2020). Accurate classification of cherry fruit using deep CNN based on hybrid pooling approach. *Postharvest Biology and Technology*, *166*, 111204.

[20] Rawat, S. S., Bisht, A., & Nijhawan, R. (2019 Nov). A deep learning based CNN framework approach for Plankton Classification. In 2019 Fifth International Conference on Image Information Processing (ICIIP) (pp. 268–273). IEEE

[21] Gupta, S., Panwar, A., Goel, S., Mittal, A., Nijhawan, R., & Singh, A. K. (2019 Dec). Classification of lesions in retinal fundus images for diabetic retinopathy using transfer learning. In 2019 International Conference on Information Technology (ICIT) (pp. 342–347). IEEE.

Preeti Malik*, Ashwini Kumar Singh, Rohit Nautiyal, Swati Rawat

Mapping AICTE cybersecurity curriculum onto CyBOK: a case study

Abstract: Several frameworks for cybersecurity education and professional development have been developed to make it easier for learners and educators to understand the various cybersecurity knowledge areas. The cybersecurity body of knowledge (CyBOK) is one of the most essential frameworks. We implemented the CyBOK mapping framework on All India Council for Technical Education-approved cybersecurity curriculum to map the curriculum with the related knowledge area in the framework.

Keywords: mapping framework, cybersecurity, higher education, curriculum, CyBOK, AICTE

1 Introduction

The Internet has become an important component of our daily lives. It has changed the way we communicate, meet new people, share information, play games, and shop. They have an impact on almost every area of our daily lives. Education is a key area for disseminating information on cybercrime prevention, and students can work as a force multiplier to help develop an ecosystem for cybersecurity and cybercrime prevention. We are practically connected to billions of online users all over the world through cyberspace.

Cybercrimes, particularly against women and children, such as cyberstalking, cyberbullying, cyber-harassment, child pornography, and rape content, are on the rise as people use the Internet more. It is critical to follow simple guidelines to keep secure online. However, the elementary understanding on which the cybersecurity domain is being established is still disbanded, and as a result, it is difficult to map consistent paths of progression through the subject. Moreover, cybersecurity has not been categorized as a distinct profession, but more of a group of related professions that have interlinking foundational concepts [1]. As a result, there are more professionals who consider themselves as cybersecurity experts but with limited cybersecurity skills or skills that do not meet the needs of the market. This is due to

*Corresponding author: **Preeti Malik,** Graphic Era deemed to be University, Dehradun, India, e-mail: preetishivach2009@gmaiul.com
Ashwini Kumar Singh, Department of Informatics, King's College London, London, United Kingdom
Rohit Nautiyal, University of Surrey, United Kingdom
Swati Rawat, Quantum University, Roorkee, India

https://doi.org/10.1515/9783110766745-007

lack of standard and clear learning pathways in the area of cybersecurity. Standards of confidentiality, integrity, and availability are essential not only for physical items but also for those items which are available in cyberspace.

In today's cyber-dominated era, security is not limited to only those things which are visible or in front of us. Rather, the scope of cybersecurity increases continuously [2]. The cybersecurity-related measures taken at this time are going to have far-reaching consequences. Today, cybersecurity has access to every small and big net-powered device, information, data, and other materials that not only help in human development but also have the ability to influence it. It is not limiting to some algorithms and protocols rather "a computing-based discipline involving technology, people, information, and processes to enable assured operations in the context of adversaries. It involves the creation, operation, analysis, and testing of secure computer systems" [3].

At present, various universities that offer cybersecurity have their own curriculum and measure progress with their own standards. Therefore, there is a need to bring all cybersecurity concepts together and standard the requirements needed for cybersecurity market. As a result, in the year 2017, National Cyber Security Programme, UK, recognized this essential demand and took actions to launch a cybersecurity body of knowledge (CyBOK) project (www.CyBOK.org) with the world-renowned cybersecurity professional to develop comprehensive curricular support and direction in cybersecurity education with an ultimate aim of providing a foundation for the development of the cybersecurity profession. The project's fundamental aim is to codify the cybersecurity knowledge which emphasizes the profession. In addition, the project will enable the development of learning and career pathways, curricula and professional training in the domain of cybersecurity [4]. CyBOK like other bodies of knowledge (e.g., SWEBOK [5]) is the knowledge that codifies already existing concepts in literature such as textbooks and academic paths of progression. Previous work has focused on outlining the ways in which CyBOK could be used to meet the needs of various stakeholders. For example, how institutions can use CyBOK to create relevant cybersecurity courses and fetch the required knowledge by going through the CyBOK knowledge areas. Author in [6] analyzed how a change in certification is applicable to maintain the curriculum. The absence of any established and unanimous framework makes the challenges of existing security even more complex [7].

In this chapter, we present a mapping based on the CyBOK for undergraduate program curriculum approved by All India Council for Technical Education (AICTE) and to show the mapping result to see the coverage of UK's CyBOK's knowledge area.

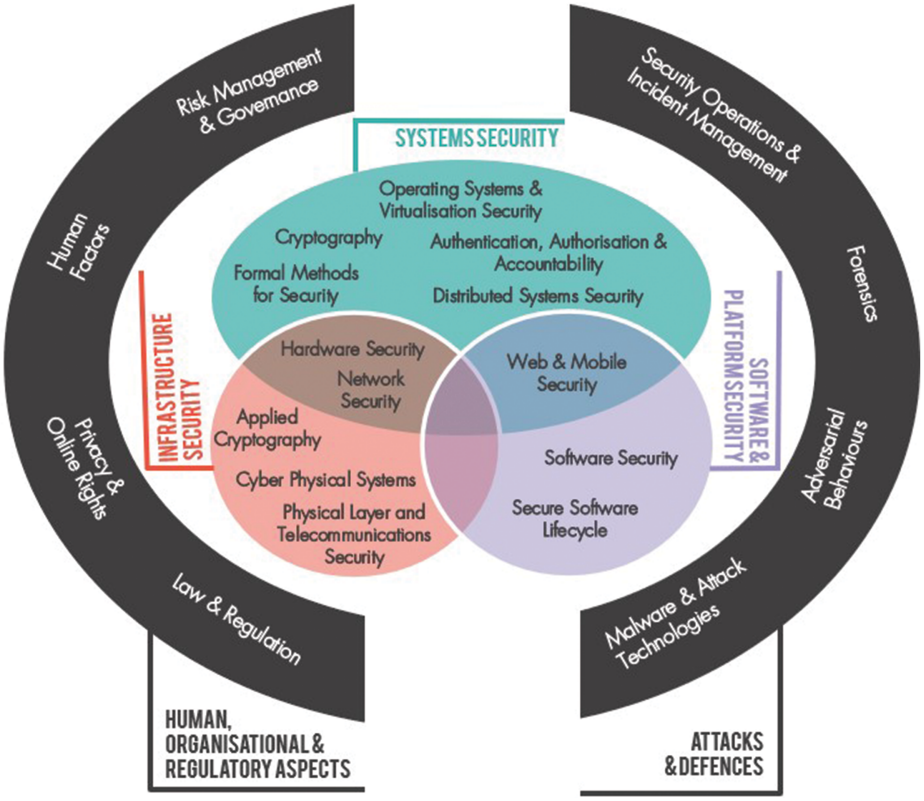


Figure 1: Official diagram of CyBOK (www.CyBOK.org).

2 What is body of knowledge (BOK)

Body of knowledge (BOK) is best stated by "structured knowledge that is used by members of a discipline to guide their practice or work." or "The prescribed aggregation of knowledge in a particular area an individual is expected to have mastered to be considered or certified as a practitioner." Another definition of BOK is "A BOK is a term used to rep- resent the complete set of concepts, terms, and activities that make up a professional Domain. It encompasses the core teachings, skills, and research in a field or industry."

Body of knowledge (BOK) is best stated by "(1) structured knowledge that is used by members of a discipline to guide their practice or work." (2) "The prescribed aggregation of knowledge in a particular area an individual is expected to have mastered to be considered or certified as a practitioner." Another definition of BOK is by "A BOK is a term used to represent the complete set of concepts, terms,

and activities that make up a professional Domain. It encompasses the core teachings, skills, and research in a field or industry [8]."

2.1 Cybersecurity initiatives by All India Council for Technical Education (AICTE)

Under the Department of Higher Education, the AICTE is a statutory entity and a national-level council for technical education. The AICTE, which began as an advisory organization in November 1945 and was later given statutory status by an Act of Parliament in 1987, is responsible for the appropriate planning and coordinated development of India's technical and management education systems.

In order to raise cybersecurity awareness among citizens and undergraduate students, the AICTE launched the cybersecurity initiative. As a result, the AICTE has developed a syllabus for undergraduate students in order to spread cybersecurity education in India [11].

2.2 Foundation of CyBOK in national certification program for academic degrees in cybersecurity in the United Kingdom

The degree certification program is one of a number of initiatives spearheaded by the UK's NCSC and its government partners in UK universities to address the knowledge, skills, and capacity required for cybersecurity research and education. The government has commissioned a project to establish the core knowledge upon which the discipline is founded in order to explore more options for future cybersecurity professionals. A team of UK academics, led by Professor Awais Rashid of Bristol University, is developing the CyBOK in conjunction with the national and worldwide cybersecurity sector. The CyBOK project was launched as part of the UK's National Cyber Security Programme to identify and explain foundational cybersecurity knowledge. Figure 1 shows the CyBOK and its knowledge areas.

This research is being carried out by a group of UK academics led by the University of Bristol, with the help of authors and reviewers from around the world. Initially, 19 cybersecurity knowledge areas were identified, which were divided into five categories: systems security, infrastructure security, software and platform security, human, organizational, and regulatory factors, and attacks and defenses. CyBOK passed with flying colors throughout the next few years, increasing from 19 to 21 knowledge areas. In order to meet the requirements of NCSC degrees, UK universities are increasingly turning to the CyBOK to define the content of cybersecurity courses. Depending on the subject paths, this could result in different "flavors" of certified degrees [9].

3 CyBOK mapping framework

The framework is based on the UK's NCSC certification requirements grounded upon the knowledge areas of the CyBOK. The framework is intended to serve as a guide for higher education institutions seeking to offer an NCSC-certified Master's program in the United Kingdom. Framework offers a step-by-step method with supplementary resources to help with each step.

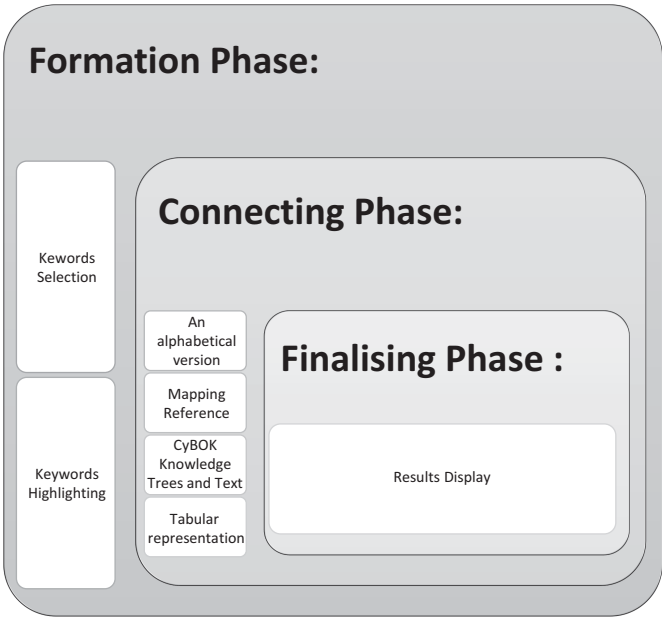


Figure 2: Phases of CyBOK mapping framework.

3.1 Mapping resources

To help explain the mapping process, this document draws upon the following materials:

1) An alphabetical version of the CyBOK's knowledge areas indicative material from the NCSC certification document
2) CyBOK mapping reference
3) CyBOK knowledge trees; CyBOK knowledge area text
4) Tabular representation of CyBOK's broad categories, knowledge areas, and their descriptions

I. **An alphabetical version of the CyBOK knowledge areas indicative material**
   It is a matrix representation of the NCSC certification document's indicative contents in alphabetical order.
II. **CyBOK mapping reference**
   It serves as a reference tool wherein CyBOK common cybersecurity words can be found. There are more than 10,000 popular cybersecurity phrases and highlighted where they could be found in the CyBOK knowledge categories.
III. **CyBOK knowledge trees**
   Knowledge of the CyBOK trees are flowchart-like tree structures in which the root node signifies a CyBOK knowledge area, each internal node signifies a topic, and its subnodes are indicative materials.
IV. **Tabular representation of CyBOK broad categories, knowledge areas and their descriptions**
   It is a matrix that shows the CyBOK broad categories and knowledge domains, as well as their descriptions. It will offer you a general overview of which knowledge areas belong to which categories, as well as describe the key aspects of each CyBOK knowledge area.

3.2 Mapping process by using the CyBOK mapping framework and mapping resources

The mapping process constitutes three phases as shown in Figure 2.

3.2.1 Formation phase

List the most particular term or group of keywords linked with the specified module during the formation phase. Do not be concerned about keyword redundancy. The goal is to compile the most comprehensive list possible. Your list could be made up of a single word, a short sentence, or a group of keywords. Highlight the keywords (or a group of keywords) that you have discovered.

3.2.2 Connecting phase

Using the tools in the CyBOK mapping resources, search for those highlighted keywords or a group of keywords. From A through E, there are five steps in this phase.

Step A: Mapping with an alphabetical version of the CyBOK's knowledge categories from the NCSC's certification document.

Step B: Mapping with the CyBOK mapping reference 1.1.

Step C: For all the recorded keywords or a group of keywords obtained by CyBOK mapping reference 1.1, complete the missing subjects and indicative material from CyBOK knowledge trees.
Step D: Using CyBOK knowledge trees to map.
Step E: Using the tabular representation of CyBOK's broad categories, knowledge domains, and their descriptions, fill in the remaining missing keywords.

3.2.3 Finalizing phase

The results of the mapping process are moved from various tables to the final table in this phase.

4 Mapping AICTE curriculum for undergraduate program

The curriculum was taken from [11]. Table 1 shows results of mapping the AICTE curriculum for UG programs onto CyBOK.

Table 1: Results of mapping.

S. no.	CyBOK Broad category	CyBOK KA	Topic	Indicative material	Keyword/set of keywords/ course keywords	AICTE model curriculum of courses at UG level in emerging areas
1	CI Human, organizational and regulatory aspects Infrastructure security	CI RMG NS	**	**	"Essential Terminologies: CIA, Risks, Breaches, Threats, Attacks, Exploits. Information Gathering (Social Engineering, Foot Printing & Scanning). Open Source/Free/Trial Tools: nmap, zenmap, Port Scanners, Network scanners."	UNIT 1: Cybersecurity concepts (2 h)

Table 1 (continued)

S. no.	CyBOK Broad category	CyBOK KA	Topic	Indicative material	Keyword/set of keywords/ course keywords	AICTE model curriculum of courses at UG level in emerging areas
2	Infrastructure security Systems security Systems security	NS C, AAA	**	**	"Introduction to Cryptography, Symmetric key Cryptography, Asymmetric key Cryptography, Message Authentication, Digital Signatures, Applications of Cryptography. Overview of Firewalls- Types of Firewalls, User Management, VPN Security, Security Protocols: - security at the Application Layer- PGP and S/MIME, Security at Transport Layer-SSL and TLS, Security at Network Layer-IPSec. Open Source/ Free/ Trial Tools: Implementation of Cryptographic techniques, OpenSSL, Hash Values Calculations MD5, SHA1, SHA256, SHA 512, Steganography (Stools) "	UNIT II: Cryptography and cryptanalysis (4 h)

Table 1 (continued)

S. no.	CyBOK Broad category	CyBOK KA	Topic	Indicative material	Keyword/set of keywords/ course keywords	AICTE model curriculum of courses at UG level in emerging areas
3	Systems security Infrastructure security	OSV NS	**	**	"Introduction to System Security, Server Security, OS Security, Physical Security, Introduction to Networks, Network packet Sniffing, Network Design Simulation. DOS/ DDOS attacks. Asset Management and Audits, Vulnerabilities and Attacks. Intrusion detection and Prevention Techniques, Host based Intrusion prevention Systems, Security Information Management, Network Session Analysis, System Integrity Validation. Open Source/ Free/ Trial Tools: DOS Attacks, DDOS attacks, Wireshark, Cain & abel, iptables/ Windows Firewall, snort, suricata, fail2ban"	Unit III: Infrastructure and network security (6 h)

Table 1 (continued)

S. no.	CyBOK Broad category	CyBOK KA	Topic	Indicative material	Keyword/set of keywords/ course keywords	AICTE model curriculum of courses at UG level in emerging areas
4	Software and platform security Systems security	WAM AAA	**	**	"Internet Security, Cloud Computing &Security, Social Network sites security, Cyber Security Vulnerabilities-Overview, vulnerabilities in software, System administration, Complex Network Architectures, Open Access to Organizational Data, Weak Authentication, Authorization, Unprotected Broadband communications, Poor Cyber Security Awareness. Cyber Security Safeguards- Overview, Access control, IT Audit, Authentication. Open Web Application Security Project (OWASP), Web Site Audit and Vulnerabilities assessment. Open Source/ Free/Trial Tools: WinAudit, Zap proxy (OWASP), burp suite, DVWA kit".	Unit IV: Cybersecurity vulnerabilities and safeguards (8 h)

Table 1 (continued)

S. no.	CyBOK Broad category	CyBOK KA	Topic	Indicative material	Keyword/set of keywords/ course keywords	AICTE model curriculum of courses at UG level in emerging areas
5	Attacks and defenses *Systems security*	MAT OSV	**	**	"Explanation of Malware, Types of Malware: Virus, Worms, Trojans, Rootkits, Robots, Adware's, Spywares, Ransom wares, Zombies etc., OS Hardening (Process Management, Memory Management, Task Management, Windows Registry/services another configuration), Malware Analysis. Open Source/ Free/Trial Tools: Antivirus Protection, Anti Spywares, System tuning tools, Anti Phishing."	Unit V: Malware (8 h)
6	Software and platform security	WAM	**	**	"Biometrics, Mobile Computing and Hardening on android and ios, IOT Security, Web server configuration and Security. Introduction, Basic security for HTTP Applications and Services, Basic Security for Web Services like SOAP, REST etc., Identity Management and Web Services, Authorization Patterns, Security Considerations, Challenges. Open Source/ Free/Trial Tools: adb for android, xcode for ios, Implementation of REST/ SOAP web services and Security implementations."	Unit VI: Security in evolving technology (8 h)

Table 1 (continued)

S. no.	CyBOK Broad category	CyBOK KA	Topic	Indicative material	Keyword/set of keywords/ course keywords	AICTE model curriculum of courses at UG level in emerging areas
7	Human, organizational and regulatory aspects Attacks and defenses	LR F	**	**	"Introduction, Cyber Security Regulations, Roles of International Law, the state and Private Sector in Cyberspace, Cyber Security Standards. The INDIAN Cyberspace, National Cyber Security Policy 2013. Introduction to Cyber Forensics, Need of Cyber Forensics, Cyber Evidence, Documentation and Management of Crime Sense, Image Capturing and its importance, Partial Volume Image, Web Attack Investigations, Denial of Service Investigations, Internet Crime Investigations, Internet Forensics, Steps for Investigating Internet Crime, Email Crime Investigations. Open Source/ Free/ Trial Tools: Case Studies related to Cyber Law, Common Forensic Tools like dd, md5sum, sha1sum, Ram dump analysis, USB device"	Unit VII: Cyber laws and forensics (9 h)

The knowledge areas are referred to by the acronyms shown further. The acronyms are explained in more detail as follows [10]:

- AAA - Authentication, Authorisation & Accountability
- AB - Adversarial behaviors
- AC - Applied cryptography
- C - Cryptography
- CI - CyBOK introduction
- CPS - Cyber-physical systems security
- DSS - Distributed systems security

- F - Forensics
- FMS - Formal methods for security
- HF - Human factors
- HS - Hardware security
- LR - Law and regulation
- MAT - Malware and attack technology
- NS - Network security
- OSV - Operating systems and virtualization
- PLT - Physical layer and telecommunications security
- POR - Privacy and online rights
- RMG - Risk management and governance
- SOIM - Security operations and incident management
- SS - Software Security SSL Secure Software Lifecycle
- WAM - Web and mobile security

5 Results and discussion

Following the successful application of the CyBOK mapping process in AICTE cybersecurity modules, the following noteworthy findings were made:

- The following graphs illustrate the coverage of the CyBOK. As demonstrated in Figure 3, the AICTE cybersecurity module covers network security to the greatest extent possible. Although we have not done a deeper level mapping of CyBOK, the findings show that each CyBOK knowledge area covered by the given module is clearly covered.
- Some of the knowledge areas are not covered by the CyBOK knowledge areas, as clearly seen in the graph.
- The AICTE courses covered all four broad categories of the CyBOK, as shown in Figure 4. Although the knowledge area has the least scope, the module has covered all of the broad categories.

6 Conclusion

In this research, we used the CyBOK mapping framework to examine the coverage of CyBOK and how the curriculum fits into an international framework on an AICTE-approved cybersecurity course. The coverage of the CyBOK knowledge area and the CyBOK broad category on the module chosen for implementation is shown in this study. By using the UK's mapping approach supplied by CYBOK, it gives us a direction that AICTE-approved cybersecurity curriculum fits in the international curriculum.

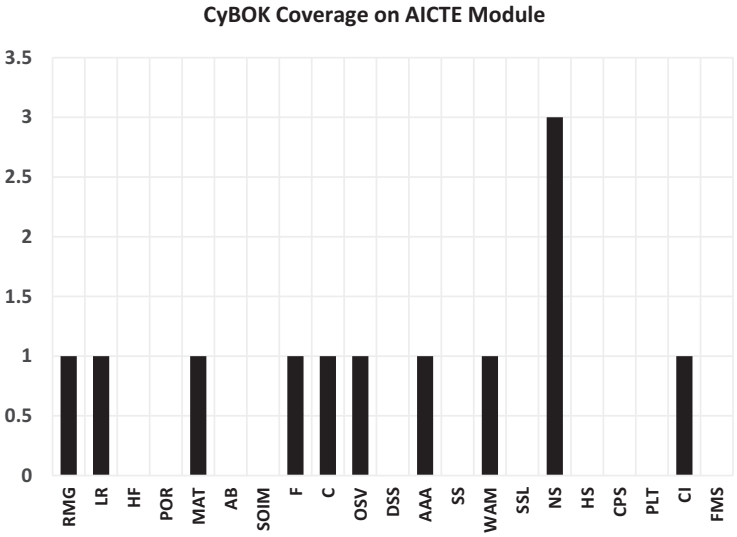


Figure 3: CyBOK knowledge area coverage on AICTE cybersecurity module.

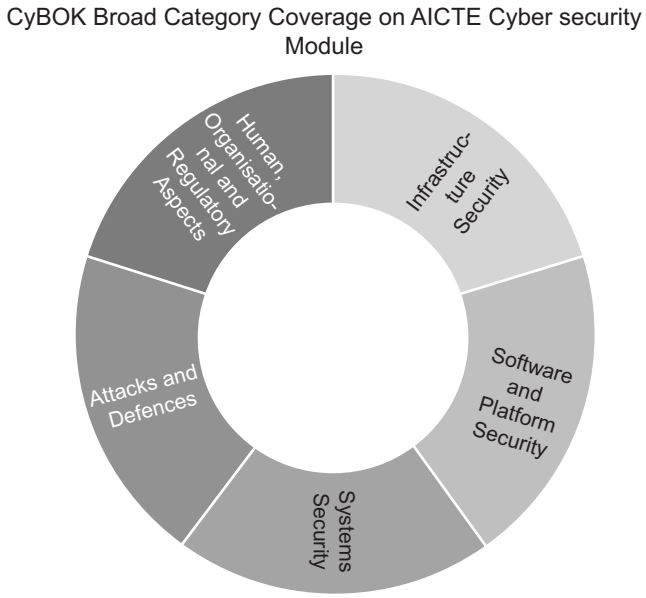


Figure 4: CyBOK broad category coverage on AICTE cybersecurity module.

References

[1] Conklin, W. A., & Bishop, M. (2018). Contrasting the CSEC 2017 and the CAE designation requirements. In Proc. 2018 51st Hawaii Int. Conf. System Sciences (pp. 2435–2441).

[2] Rashid, A., Danezis, G., Chivers, H., Lupu, E., Martin, A., Lewis, M., & Peersman, C. (2018). Scoping the cyber security body of knowledge. In IEEE security \& privacy.

[3] ACM, IEEE, AIS, and IFIP. (2017). Cybersecurity Curricula 2017. https://cybered.hosting.acm.org/wp.

[4] Rashid, A., Chivers, H., Lupu, E., Martin, A., & Schneder, S. (2021). The cyber security body of knowledge version 1.1.0.

[5] Bourque, P., & Fairley, R. E. (eds.). 2014. Guide to the Software Engineering Body of Knowledge (SWEBOK). In Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.swebok.org.

[6] Knapp, K. J., Maurer, C., & Plachkinova, M. (2017). Maintaining a cybersecurity curriculum: Professional certifications as valuable guidance. *Journal of Information Systems Education 28*(2), 101–113.

[7] Hallett, J., Larson, R., & Rashid, A. (2018) Mirror, mirror, on the wall: What are we teaching them all? Characterising the focus of cybersecurity curricular frameworks. In Proc. Adv. Security Educ. Workshop (ASE), USENIX Security Symp.

[8] Oliver, G. R. (2012). Foundations of the Assumed Business Operations and Strategy Body of Knowledge (BOSBOK): An Outline of Shareable Knowledge (pp. 3).

[9] Nautiyal, L., Rashid, A., Hallett, J., Shreeve, B., Michael, K., Chris, E., & Catherine, H. (2022). The united kingdom's cyber security degree certification program: A cyber security body of knowledge case study. *IEEE Security & Privacy*, *20*(1), 87–95.

[10] www.cybok.org

[11] https://www.aicte-india.org

Index

https://doi.org/10.1515/9783110766745-008

De Gruyter series on the applications of mathematics in engineering and information sciences

Already published in the series

Volume 14: Multiple Criteria Decision-Making Methods. Applications for Managerial Discretion
Mohini Agarwal, Adarsh Anand, Deepti Aggrawal
ISBN 978-3-11-074356-2, e-ISBN (PDF) 978-3-11-074363-0,
e-ISBN (EPUB) 978-3-11-074374-6

Volume 13: Integral Transforms and Applications
Nita H. Shah, Monika K. Naik
ISBN 978-3-11-079282-9, e-ISBN (PDF) 978-3-11-079285-0,
e-ISBN (EPUB) 978-3-11-079292-8

Volume 12: Noise Filtering for Big Data Analytics
Souvik Bhattacharyya, Koushik Ghosh (Eds.)
ISBN 978-3-11-069709-4, e-ISBN (PDF) 978-3-11-069721-6,
e-ISBN (EPUB) 978-3-11-069726-1

Volume 11: Artificial Intelligence for Signal Processing and Wireless Communication
Abhinav Sharma, Arpit Jain, Ashwini Kumar Arya, Mangey Ram (Eds.)
ISBN 978-3-11-073882-7, e-ISBN (PDF) 978-3-11-073465-2,
e-ISBN (EPUB) 978-3-11-073472-0

Volume 10: Meta-heuristic Optimization Techniques. Applications in Engineering
Anuj Kumar, Sangeeta Pant, Mangey Ram, Om Yadav (Eds.)
ISBN 978-3-11-071617-7, e-ISBN (PDF) 978-3-11-071621-4,
e-ISBN (EPUB) 978-3-11-071625-2

Volume 9: Linear Integer Programming. Theory, Applications, Recent Developments
Elias Munapo, Santosh Kumar
ISBN 978-3-11-070292-7, e-ISBN (PDF) 978-3-11-070302-3
e-ISBN (EPUB) 978-3-11-070311-5

Volume 8: Mathematics for Reliability Engineering. Modern Concepts and Applications
Mangey Ram, Liudong Xing (Eds.)
ISBN 978-3-11-072556-8, e-ISBN (PDF) 978-3-11-072563-6
e-ISBN (EPUB) 978-3-11-072559-9

Volume 7: Mathematical Fluid Mechanics. Advances on Convection Instabilities and Incompressible Fluid Flow
B. Mahanthesh (Ed.)
ISBN 978-3-11-069603-5, e-ISBN (PDF) 978-3-11-069608-0
e-ISBN (EPUB) 978-3-11-069612-7

www.degruyter.com

Volume 6: Distributed Denial of Service Attacks. Concepts, Mathematical and Cryptographic Solutions
Rajeev Singh, Mangey Ram (Eds.)
ISBN 978-3-11-061675-0, e-ISBN (PDF) 978-3-11-061975-1
e-ISBN (EPUB) 978-3-11-061985-0

Volume 5: Systems Reliability Engineering. Modeling and Performance Improvement
Amit Kumar, Mangey Ram (Eds.)
ISBN 978-3-11-060454-2, e-ISBN (PDF) 978-3-11-061737-5
e-ISBN (EPUB) 978-3-11-061754-2

Volume 4: Systems Performance Modeling
Adarsh Anand, Mangey Ram (Eds.)
ISBN 978-3-11-060450-4, e-ISBN (PDF) 978-3-11-061905-8
e-ISBN (EPUB) 978-3-11-060763-5

Volume 3: Computational Intelligence. Theoretical Advances and Advanced Applications
Dinesh C. S. Bisht, Mangey Ram (Eds.)
ISBN 978-3-11-065524-7, e-ISBN (PDF) 978-3-11-067135-3
e-ISBN (EPUB) 978-3-11-066833-9

Volume 2: Supply Chain Sustainability. Modeling and Innovative Research Frameworks
Sachin Kumar Mangla, Mangey Ram (Eds.)
ISBN 978-3-11-062556-1, e-ISBN (PDF) 978-3-11-062859-3,
e-ISBN (EPUB) 978-3-11-062568-4

Volume 1: Soft Computing. Techniques in Engineering Sciences
Mangey Ram, Suraj B. Singh (Eds.)
ISBN 978-3-11-062560-8, e-ISBN (PDF) 978-3-11-062861-6,
e-ISBN (EPUB) 978-3-11-062571-4