IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 7 December 2022, accepted 28 December 2022, date of publication 5 January 2023, date of current version 11 January 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3234434*

RESEARCH ARTICLE

# SDOT: Secure Hash, Semantic Keyword Extraction, and Dynamic Operator Pattern-Based Three-Tier Forensic Classification Framework

**D. PAUL JOSEPH[1], (Graduate Student Member, IEEE), AND P. VISWANATHAN[2], (Senior Member, IEEE)**

[1]School of Information Technology and Engineering, Vellore Institute of Technology, Vellore 632014, India
[2]School of Computer Science and Engineering, Vellore Institute of Technology, Vellore 632014, India

Corresponding author: P. Viswanathan (pviswanathan@vit.ac.in)

**ABSTRACT** Most traditional digital forensic techniques identify irrelevant files in a corpus using keyword search, frequent hashes, frequent paths, and frequent size methods. These methods are based on Message Digest and Secure Hash Algorithm-1, which result in a hash collision. The threshold criteria of files based on frequent sizes will lead to imprecise threshold values that result in an increased evaluation of irrelevant files. The blacklisted keywords used in forensic search are based on literal and non-lexical, thus resulting in increased false-positive search results and failure to disambiguate unstructured text. Due to this, many extraneous files are also being considered for further investigations, exacerbating the time lag. Moreover, the non-availability of standardized forensic labeled data results in ($O(2^n)$) time complexity during the file classification process. This research proposes a three-tier Keyword Metadata Pattern framework to overcome these significant concerns. Initially, Secure Hash algorithm-256 hash for the entire corpus is constructed along with custom regex and stop-words module to overcome hash collision, imprecise threshold values, and eliminate recurrent files. Then blacklisted keywords are constructed by identifying vectorized words that have proximity to overcome traditional keyword search's drawbacks and to overcome false positive results. Dynamic forensic relevant patterns based on massive password datasets are designed to search for unique, relevant patterns to identify the significant files and overcome the time lag. Based on tier-2 results, files are preliminarily classified automatically in O(log n) complexity, and the system is trained with a machine learning model. Finally, when experimentally evaluated, the overall proposed system was found to be very effective, outperforming the existing two-tier model in terms of finding relevant files by automated labeling and classification in O(nlog n) complexity. Our proposed model could eliminate 223K irrelevant files and reduce the corpus by 4.1% in tier-1, identify 16.06% of sensitive files in tier-2, and classify files with 91% precision, 95% sensitivity, 91% accuracy, and 0.11% Hamming loss compared to the two-tier system.

**INDEX TERMS** Digital forensics, disc forensics, forensic data classification, metadata, pattern, blacklisted keywords.

## I. INTRODUCTION

Digital forensics(DF) is concerned with digital device seizure, analysis, and preservation. When a cybercrime, such as hacking, internet fraud, or identity theft, is detected, the devices involved are seized, preserved, and sent for forensic analysis. As a result, the cyber world is inextricably linked to the digital forensics domain, and the two go hand in hand. Digital forensics encompasses many disciplines, including disc forensics, network forensics, memory

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

forensics, cloud forensics, database forensics, multimedia forensics, and mobile forensics [1]. In general, when a system or any digital device is compromised or attacked, it is seized, which is substantiated by taking a bit-by-bit image copy followed by hashing, resulting in twice the size of the actual data. The number of devices seized for analysis is proportional to the number of cases registered, as is the rationale for the pending cases. According to national crime records bureau statistics from 2014 to 2018, there was a four-fold increase in pending cyber cases, with identity theft and transmission of obscene content factoring for a higher percentage [2]. Furthermore, according to the Federal Bureau of Investigation Regional Computer Forensics Laboratories (CFLs) annual report [FBI08], it processed more data than the previous year [3]. These statistics demonstrate that massive data is unavoidable.

Due to the limited availability of forensic resources and the inability to distinguish between relevant and irrelevant forensic files, investigations have a significant time delay [4]. Technically, files that do not aid in forensic investigations are deemed irrelevant, and these files encompass software installation files, operating system-related files, and standard temporary files. However, relevant or sensitive files are relevant to the investigations and therefore reveal personal information that includes a person's name, address, password, emails, bank information, and so on. From the perspective of a forensic investigator, it is facile to differentiate the relevant and irrelevant files in millions of RDC (Real Drive Corpus) files.

National Software Reference Library (NSRL) introduced a known-good-hash set with Message Digest(MD5) and Secure Hash Algorithm(SHA1) methods to eliminate irrelevant files in a corpus [5]. These hash sets are also known as whitelisted hashes and are used to eliminate files matched against known hashes. Hash values of any entity in NSRL using MD5 and SHA1 are calculated by "(1)."

For example, refer "(1)" to evaluate the hash value of a file $F_i$ and $S_i$ in NSRL.

$$h(F_i), (S_i) = \left( \sum_{i=0}^{n-1} a^{n-1-i} f_i, s_i \right) \bmod p \tag{1}$$

then the condition to eliminate matched file is given in "(2)"

$$\forall\, (F_i,\ S_i) \begin{cases} if & h\,(F_i) = h\,(S_i) \ \rightarrow Del\ (S_i) \\ if & h\,(F_i) \neq h\,(S_i) \ \rightarrow Add\ (S_i) \end{cases} \tag{2}$$

where $f_i$ is the predefined hash in NSRL and $s_i$ is the computed hash for new files in RDC, $a$ is a parameter in the rolling hash function larger than $|\Sigma|$ and $p$ is a parameter in the rolling hash function that should be a big prime.

The problems with this approach are threefold: The first point is that it is static, and the hash data set must be updated periodically. Second, even though NSRL identified over 202 million uninteresting hashes globally as of December 2021 [5], millions of irrelevant files remain unidentified and processed, further consuming expensive computational resources [6]. Thirdly, MD5 and SHA-1 are highly prone to active adversaries attack and are not recommended for forensic activities [7], [8]. To solve hashing issues, Forensic hash matching using side information is implemented by [9] and evaluated denseness index to characterize hash data by adding file sizes and pre-hashes to reduce NSRL hash comparison. The denseness index is calculated using "(3)."

$$DI_{H(n)} = \frac{100 D_H\ ((n-1)\, s, ns)}{s} \tag{3}$$

where $s$ represents file size and $n$ represents slots. The problem with "(3)" is that it reduces file comparison by considering side-information alone, such as considering file sizes within the range, thereby resulting in an imprecise threshold value. Frequent hashes, frequent paths, frequent bottom-level pairs, frequent sizes, clustered creation times, contextually irrelevant files, and known irrelevant extension techniques are evaluated by [10] to identify and eliminate the most irrelevant files. According to [10], if $y = H(x)$ where $x \in R$, $y \in [0, 9]$, *and* $|x|$ is the floor function, then the criteria to eliminate irrelevant forensic files is given in "(4)."

$$\prod_{0}^{max} x = \begin{cases} -1, & \text{if } x \in X \\ +1, & \text{otherwise} \end{cases} \tag{4}$$

The problems with the above approach are: these techniques are confined to RDC corpora alone but not applicable to other drives. Secondly, usage of MD5 and SHA-1 hashes in this work results in a hash collision attack [11]. Due to the function defined in "(4)", many files with altered extensions could not be validated and eliminated, thus resulting in false negatives. To overcome this issue, a hybrid methodology is evaluated with redefined parametric threshold values to eliminate far more forensically irrelevant files in RDC by [6], yet time complexity remains exponentially high. Another method to identify the file's interestingness or relevancy is by considering metadata properties, and in specific, a file's magic number is considered. A colossal list of magic numbers for most file signatures to be used in cross-validation of any file is developed by [12]. A combination of file signature and keyword search is developed based on these file signatures [13]. Since keywords are selected based on the repetition of words in the corpus, many sensitive words that appeared a few times in the corpus are unrecognized. Further, no developments are made in the file signature. To solve this issue, a digital forensic toolkit is developed by [14] for extracting file metadata and generating timeline, but this toolkit extracts information based on file attributes, but not on the magic number of a file, which results in bypassing of files with altered extensions. With the same file signatures, [15] analyzed anti-forensic capabilities by modifying a file's magic number and demonstrated that such altering significantly results in a damaged file and misleading the investigators. To solve this issue, [16] developed a forensic toolkit based on a magic number extension checker, but this toolkit is confined to limited pre-defined modules.

Apart from the techniques mentioned above, keyword search, pattern matching, and trained classifiers in machine learning models have been implemented in the past to find sensitive or relevant files. The first two methods necessitate manual intervention to compile keywords and patterns, culminating in exponential time complexity [17], [18]. The latter approach includes (1)-clustering algorithms to cluster similar documents based on relevancy, date of creation, file size, and Self Organizing map (SOM) [19], [20], [21]. (2)-topic classification models such as Latent Dirichlet allocation (LDA) and Latent semantic analysis (LSA) to detect latent topics so that an investigator could indeed flag a file as sensitive or not by perceiving the latent topics [22], [23], [24]. These approaches work with unsupervised data and extract latent topics in addition to the central theme of a document in a corpus. The major problem is that even after the topics are extracted, manual intervention is required to detect or classify whether a file is sensitive or not, as well as topic instability [25]. (3)-Furthermore, data classification algorithms in machine learning such as Naive Bayes, support vector machines, and decision trees are used in Digital forensics [26], [27] to classify the file as sensitive or not [28], [29]. The problem with these approaches is that they only work on supervised data, which must be labeled according to the forensics domain such that Data classification is performed based on data labeling. Another issue is that one must have extensive knowledge of the forensic domain to train the data, which would otherwise result in a significant loss.

This research proposes a ''three-tier Keyword Metadata Pattern'' framework to overcome these significant concerns. SHA-256 hash for the entire corpus is constructed with custom regex and stop-words modules to overcome hash collision, approximate threshold values, and eliminate recurrent files in tier-1. In stage 1 of tier 2, the new blacklisted keywords dataset is constructed using the word-to-vector and latent Dirichlet allocation algorithm. The metadata module of stage 2 is proposed to overcome false positive results in an existing system. The unique pattern module at stage 3 is designed to search for dynamic, unique, relevant patterns to identify the significant forensic relevant files and overcome the time lag.

This paper is organized in the following manner. Section 2 discusses related work, while Section 3 describes the proposed three-tier KMP classifier. Section 4 evaluates the proposed approach and presents the results, and finally, Section 5 summarises the conclusion and effectiveness of the proposed classifier.

## II. RELATED WORK

Technology-aided forensic investigation advancements use machine learning, artificial intelligence, and natural language processing. Finding relevant files has piqued the interest of forensic researchers, and some of the methods are discussed in this section. Using standard hash sets from NSRL and virus share to eliminate common irrelevant files has become a traditional approach in Forensic Toolkit(FTK). In forensic corpus reduction techniques, NSRL known-good-hashes, Rowe's hash, peka torrent, virus share, and OS forensic hashes are used to implement hybrid filtering techniques that use MD5 and SHA-1 hashes for each entity [6]. The major problem with the above hash dataset lies in the compressed hash function and bitwise function *f(p,q,r)*, as given in ''Fig. 1.''

$$H_i = [2^{32} \text{ X abs}(\sin(i+1))], \text{ where H\_i is additive constant}$$

$$f(p,q,r) = (p \wedge q) \vee (\neg p \,\&\, q) \text{ for } 0 \le i \le 15$$

$$f(p,q,r) = (p \wedge q) \vee (q \wedge \neg r) \text{ for } 16 \le i \le 31$$

$$f(p,q,r) = p \oplus q \oplus r \text{ for } 32 \le i \le 47$$

$$f(p,q,r) = q \oplus (p \vee \neg q) \text{ for } 48 \le i \le 63$$

**FIGURE 1.** Working of hash function and bitwise function in message digest algorithm.

Since *p,q, and r* are 0-31 bit words, the output generated from the four rounds is 128bit which is subsequently prone to known differential attack or hash collision attack in 2$\hat{}$64 rounds. Furthermore, [6] defines a set of algorithms with redefined threshold values to eliminate uninteresting files. Nevertheless, the most significant issue lies in the function *f(x)* defined with static parameters as given in ''(5)'' and ''(6).'' In ''(5),'' $\sum_{k=0}^{n} del\_size_i$ is a predefined static function where the threshold values of specific files are initialized. If any file matches according to its threshold criteria, those files are supposed to be irrelevant. For example, the threshold value for a word document is defined as $file\_ext[w_i] = 4 * 1024$ bytes, and text file as $file\_ext[t_i] = 4 * 1024$ bytes. To validate the results, we created a word document and text file with an 8-digit word, then used Huffman coding to compress the data, which resulted in a document size of 3000 bits and a text file size of 2000 bits which is 1000bits lesser than the existing. Likewise, for remaining extensions, we found many threshold values imprecisely defined that lead to false negatives. Another problem exists in $\sum_{i=0}^{n} extlist_{i,j}$ of ''(6)'' which predefines some relevant and irrelevant file extensions. Some of the irrelevant extensions predefined in [6] are [.inf, .ext, .bin, .cab, .cfg, .cpl, .cur, .drv] and relevant extensions as [.dll, .bat, .csv, .doc, .txt, .xls, .rtf, .ppt]. Suppose any file matches with the irrelevant category extension, those files are treated as an uninteresting or irrelevant category, and any file that matches the relevant category is sent for further processing. The problem with this function is that if any file's header is altered or deleted, this approach fails to identify that file as sensitive as the new extension might not be defined in their list. As a result, we observed that many significant files were removed from further investigation, raising the question of reliability. Also, as extensions are predefined based on ''(5)'' and ''(6),'' this

work can identify uninteresting files more than interesting files in RDC, as explained above. The shortcomings of these two functions can be addressed in our proposed work by using the *file_ext*($D_i$) module of the KMPT classifier, where new threshold values are defined after corpus evaluation, and a novel algorithm to detect deleted or altered file extensions are proposed in Tier-2.

$$\prod_{n}^{max} f\,(x) \;=\; \begin{cases} -1, & x < del\_size_1 |\; del\_size_2 \ldots |\; del\_size_n \\ +1 & otherwise \end{cases} \tag{5}$$

$$\prod_{x}^{n} f\,(x) \leftarrow if\;\; [X_i] \subset [extlist_i]\;\; then\; del\;\; [X_i] \tag{6}$$

Table 1 refers to the frequent notations used in this work.

**TABLE 1. Important notations used in this research.**

| Notation | Description |
|---|---|
| ASCII | American Standard Code for Information Interchange |
| BKW | Blacklisted Keyword |
| BOW | Bag-of-Words |
| CFT | Computer Forensic Tool |
| D2V | Document-to-Vector |
| DF | Digital Forensics |
| DHS | Department of Homeland Security |
| FKS | Forensic Keyword Search |
| IDF | Inverse Document Frequency |
| KMP | Keyword-Metadata-Pattern |
| LDA | Latent Dirchilet Allocation |
| MCC | Matthew's correlation coefficient |
| NLP | Natural Language Processing |
| NLTK | Natural Language Toolkit |
| NSRL | National Software Reference Library |
| RDC | Real Drive Corpus |
| SSN | Social Security Number |
| SVM | Support Vector Machine |
| TF | Term frequency |
| W2V | Word-to-Vector |

Forensic keyword search(FKS), an essential function of any forensic tool, is another approach to identifying relevant files. FKS aims to identify suspicious files in massive data by predefined keywords. For this purpose, the DHS released blacklisted keywords under eight categories widely used in CFTs apart from expert case-specific keywords. However, improperly devised keywords or keywords formulated by experts result in high false positive rate [18], [30]. Another problem with DHS keywords is that they are non-lexical; therefore, exact matches result from bypassing meaningful or related words. As a result, pipelined keyword enhancing technique is implemented in [31] by integrating seed keywords with pipelines. In their work, the authors mentioned that the most relevant words with a particular topic are identified as seed keywords. The major problem lies in determining seed words using "(7)."

$$tf\,(t_i, d_i) = \frac{f_{t_i,d_i}}{\sum_{t_i^{'} \in d_i} f_{t_i^{'},d_i}} \tag{7}$$

For any term that is least present in document(D), then "(7)" becomes

$$\prod_{t_i=0}^{\max(t_i)} \left\{ \begin{matrix} t(t_i, d_i) = 1 \text{ if } t \in D \\ t(t_i, d_i) = 0 \text{ if } t \notin D \end{matrix} \right\} \tag{8}$$

Therefore, if any *word(w)* is not frequently present in the *corpus(C)*, then that word is not treated as a seed word. Another problem lies in the identification of seed keywords using W2V given in "(9)."

$$S_c\left(D_{m,a}, D'_{m,a}\right) = \frac{\sum_{i=1}^{N} D_{m,a} D'_{m,a}}{\sqrt{\sum_{i=1}^{N} D^2_{m,a}} \sqrt{\sum_{i=1}^{N} D'^2_{m,a}}} \tag{9}$$

where $D$ is corresponding data, $D^0_{m,a}$ is initial seed keyword with topic $m$ for level 1 and $N = 1..10$.

Equation (9) identifies the top N keywords as seed keywords and then manually labels them. Therefore, the problem is that there is no guarantee that essential words must occur utmost, so many significant keywords are bypassed in this framework. Based on the ground truth dataset, in tier 2, a blacklisted keyword search module is proposed by integrating the W2V model and LDA topical modeling algorithm to overcome the above issues.

The unavailability of labeled datasets is an increasing concern in DF. Before processing the data, it should be labeled as most available data is unstructured or unsupervised [32]. The manual labeling technique is adapted and has still been used in some cases, which was quite tedious; later, with the help of feature extraction algorithms, unstructured data is converted to either semi-supervised or supervised data. As a result, text mining is gaining traction as it uses NLP to transform unstructured content into structured content [33], [34]. During data transformation, as many inconsistencies appear in the data, GapFinder is implemented for this purpose [35] which primarily focuses on extracting structured data from semi-structured data. The authors claimed in this paper that they implemented a topic classifier that uses the D2V model for word representation and the SVM model for article classification. The major problem with this approach is that it uses the D2V model for classification, which looks specifically for the headlines of articles and then classifies the data after vectorization. When we implemented the same technique, we discovered many false negatives in our work by implementing D2V on headlines alone, and how the existing system results in false negatives is given in "(10)." For example, consider an article in vector space containing a sequence of words; the context for the word $p$ is given as $P(w_t)$ with window size 2. For the given probability

$$P\left(w_j\right) = \left[\sqrt{\frac{Z\left(w_j\right)}{K}} + 1\right] . \frac{K}{w_j} \tag{10}$$

where $Z(w_j)$ is the normalized frequency of occurrence and $K$ is a scale factor. As headlines are unique to each document and do not appear elsewhere, $Z(w_j)$ will always be 1, and

the D2V model classifies the article as non-cyber security, resulting in a false negative. This is because it is optional for a subject to align with the topic headline, and we identified many such pages. To address this, the authors proposed KMPT-based D2V that performs full-text vectorization in tier-2 and SVM with linear kernel in tier-3 to reduce false negatives significantly.

## III. KEYWORD METADATA PATTERN CLASSIFIER

To address the above shortcomings, we propose a three-tier framework depicted in ''Fig. 2''. Tier 1 consists of data extraction and a pre-processing engine [36]. The pre-processing engine includes tokenization, stop words, and lemmatization modules so that the input for Tier-2 is cleaned data [33], [37]. Tier 2 includes the KMP classifier, a novel forensic text-relevant classifier system for labeling and preliminary classifying of forensically relevant data. Tier 2 identifies and classifies the most relevant files in an RDC. As previously stated, suspicious files can be of any type, including a person's name, location, account details, user Id's, passwords, SSN, credit/debit card details, mac address, IP address, email Id's, and so on [38]. Keyword or string search [1], regex search, and hash value search are used as primary sources of searching for sensitive files individually. However, in this work, we combined keyword search, magic file number search [39], and pattern search [40] as a single entity for optimal results. The preliminary classification in this phase is evaluated with the help of $bkw(D_i)$, $file\_ext(D_i)$, and $pattern(D_i)$ modules. The resultant data from tier 2 is vectorized using the D2V model and given as input to tier 3. Tier 3 includes the Linear SVM learning classifier that automatically detects and classifies forensically relevant files. Advantages of the proposed system are: better accuracy in finding and classifying forensically relevant files; less manual work during investigations; three-fold evaluation to find suspicious files to avoid false negatives; and better performance evaluation metrics like precision, recall, f1-score, accuracy, specificity, Hamming loss, and Matthew's correlation coefficient(MCC).

### A. DATASETS DESCRIPTION AND ACQUISITION

All our experimental evaluations for this research are carried out on standard datasets such as T5 Corpus (http://roussev.net/t5/t5.html), which contains 4,457files, MS-X 13 corpus (http://roussev.net/msx-13/msx-13.html) that contains 16K files, Digital corpora dataset [41], NSRL hash set [5], Computer Forensic Reference Data [41]. Apart from these standard datasets, to overcome hash collision attacks, we developed SHA-256 hash datasets with 1.05 million unique entries. Data acquisition from physical sources is made with the help of FTK Imager, as it supports Linux, mac, and windows operating systems in addition to a software write blocker mechanism that does not modify file timestamps. For data authentication, SHA-256 hash is computed for each entity in data acquisition and stored in a central repository. Security-related websites that allow scrapping their site are scrapped with the help of BeautifulSoup and pandas as per constraints mentioned in the robots.txt file. All the relevant sources used in this work can be found at ''https://github.com/pauljoseph91/KMPT.''

### B. TIER-1: DATA PRE-PROCESSING

In tier 1, we enhanced the existing pre-processing engine module, which converts raw data into the machine-understandable format by performing morpho-syntactic analysis, and inverted indexing techniques for full-text search [42]. In this work, data acquisition is performed on various sources such as personal computers, mechanical tape drives, and internet sources and then sent to pre-processing engine for data cleaning purposes. In morpho-syntactic analysis, the first step is tokenization, where each word in the corpus is converted into an individual token. After conversion, these tokens are sent to the stop words module custom-tailored for digital forensics.

Stop words are common in spoken language, defined in the nltk library. These stop-words that impede the computing process in RDC are identified with the help of Inverse-document frequency, as mentioned in

$$Idf(s_i) = log(n/m)$$

where $s_i$ is specific term in corpus(T), $n$ is document and $m$ being the frequency of a word. The process for defining a custom stopwords module is given below.

```
f_sw=['digital', 'byol', 'cots','dlp', 'edr', '
    fido', 'deploy', 'account']
stpwrd = nltk.corpus.stopwords.words('english')
stpwrd.extend(f_sw)
del_sw = [words for words in text_tokens if not
    words in stpwrd]
```

**Listing 1.** Forensic tailored stop words module.

All the stop words in $T$ are removed that are defined in $f\_stwords(T, f_s)$. Following that, we used the Lemmatization process, which considers a word's context and derives it from the root word. Even though lemmatization takes longer than stemming, in the current context of forensic analysis, this text normalization technique is highly regarded. Finally, in tier-1, after performing morpho-syntactic analysis, we indexed our data using an inverted index algorithm using NLP for full-text search with quick response [42]. The pseudo code for the tier-1 model is provided in the algorithm 1.

### C. TIER-2: KEYWORD-META-PATTERN CLASSIFIER

This section combines blacklisted keywords, metadata, and pattern searches into a single entity. We started with a ground-truth dataset of BKW defined by the DHS and forensic experts for keyword searches. These are the keywords that are presumed sensitive worldwide for surveilling terrorist activities, emails, digital chats, or any unethical or illegal activity. We then created a repository containing file extensions, associated magic numbers, and relevant ASCII codes

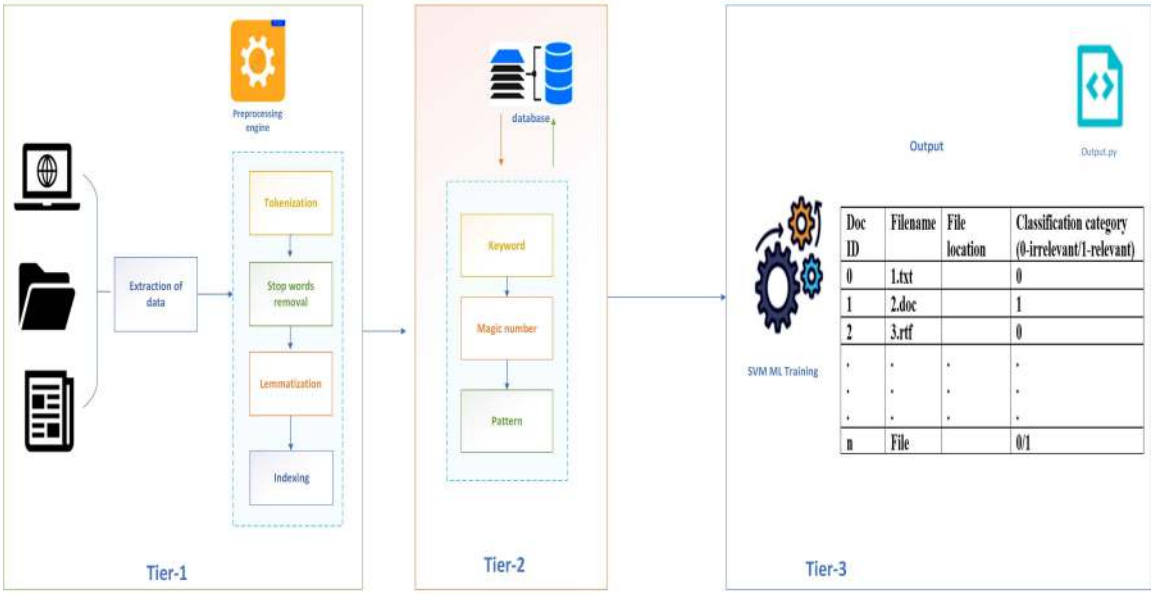


**FIGURE 2.** Proposed KMPT Three-tier framework and Work flow of Tier-2 KMPT.

for metadata search. Finally, we created a set of unique patterns to identify suspicious files at the word level for pattern search. "Fig. 3" shows the workflow of tier 2.

### 1) BLACKLISTED KEYWORD MATCH

Blacklisted keyword(BKW) match technique is used in this research to evaluate any illegal, unethical, or unlawful activities in RDC or on an individual's device. DHS keywords have two significant limitations. First, they are scant in both quantity and diversity. Second, similar words or words with proximity are not considered. For example, "terrorism" is a blacklisted word in DHS, but ISIS is a major terrorist organization that is not mentioned anywhere else in DHS.

As a consequence, many undefined sensitive keywords may slip through. Second, "site" is one of the blacklisted words in the DHS list. Whether the word "site" refers to a location or a website is unclear. This study overwhelms the dual drawback while proposing a method for identifying BKW and similar words. We used the W2V model with LDA, a topical modeling algorithm, and the flow of identifying new BKW is illustrated in "Fig. (4)."

The BKW ground truth dataset is constructed with the help of forensic experts, and the entire corpus $T$ is trained with W2V and LDA algorithm to get DHS-relevant keywords with the context. We then transformed the pre-processing data to vectors using the W2V module with the continuous skip-gram model by using Negative sampling. The vectorized words from W2V are now fed into the $LDA(V_i, K)$ model, as shown in the algorithm 2.

Furthermore, we stringently trained to find relevant words in the test data. The output of the LDA model is stored in a central repository and can be compared to any forensic dataset to identify keywords. On RDC, we compared our proposed BKW model with the DHS BKW over 25 topics, and we present results for cyber security and terrorism. Our work identified 16 keywords in cyber security topics, whereas the existing method could identify only seven keywords. Similarly, for the topic "terrorism," our proposed work identified 22 keywords, whereas existing work identified 13 words. Even though DHS keywords are the base for any keyword identification in FTK, the major problem is that investigators search these keywords literally. To overcome this issue, we proposed algorithm 2 to identify new sensitive keywords based on semantics and relevancy. When the proposed BKW identification method is compared to the existing keyword techniques, it is evident that it resulted in identifying keywords with a minimum of 40% efficiency. The comparison result is shown below.

**Topic 0: Cyber Security**

**DHS:** Malware + Virus + Trojan + Rootkit + MySQL injection + Phishing + Worm.; Identified words:7

**Proposed Work:** Zbot + Trojan + Malware + Ransomware + Backdoor + adware + spyware + rootkits + malvertising + SQL injection + Botware + Phishing + Win32.Worm + Macro virus + Logic bomb + crypt.; Identified words:16

**Topic 1: Terrorism**

**DHS:** Al Qaeda + IED + Abu Sayyaf + Hamas + FARC + Hezbollah + Tamil Tigers + PLF (Palestine liberation Front) + PLO (Palestine Liberation Organization) + Jihad + Taliban + TTP (Tehrike -e -Taliban Pakistan) + Pirates.; Identified words:13

**Proposed Work:** Al Qaeda + Boko Haram + Haqqani network + Jaish -e -Mohammad + Lashkar-e-Taiba + Al Shabab + BLO + ETLO (East Turkestan Liberation Organization) + ETIC (Information Centre) + Hamas + Hezbollah + Islamic Jihad + Jamait-e-Islami + PLF (Palestine

**Algorithm 1** Pseudo Code for Tier 1

```
Input:raw data
Output:tokens, stopwords,lemma
lexical_analysis()
def tokenization(Data, T)
begin
    read text
    initialize tok_list[]
    Assign re.findall("[\ w']+').split() to tokens
    variable //regex tokenization
    for token ∈ text do
        tok_list.append(token.text)
    end
end
def f_stwords(T, fs)
begin
    stop_words= stopwords to stop_words;
    fs=[w for w ∈ tok_list if w ∉ stop_words]
    a = []
    for w ∈ tok_list : do
        if w ∉ stop_words : then
            fs.append(w);
        end
    end
end
def lemmetization(fs, lemma):
begin
    lemmatizer = WordNetLemmatizer().
    lemma=[] fs=[w for w ∈ tok_list if w ∉
    stop_words]
    for token ∈ tok_list : do
        lemmetized_word =
        lemmatizer.lemmatize(token);
        lemma.append(lemmetized_word)
    end
end
```

liberation front) + Taliban + ISI + ISIS + TTP + Syria + Explosive + car bomb + bio warfare.; Identified words:22

### 2) FILE EXTENSION MANIPULATION

This module attempts to recognize and classify files based on changes to their magic number. Each file has its file identification number in a unique Hex form. In this module, we first passed our data set($T$) to the checker.py module, providing three checking functionality types. In this module, the first 24 bytes, along with the file name and the extension, are extracted from each file and compared to a central repository containing over 25K registered file-type extensions. Second, the same file is queried to a dynamic web server, where the list of file extensions is updated at periodic intervals. Finally, to avoid false negatives, we deployed the XXD tool [43] to confirm the file alteration. Additionally, this module identifies files with incompatible extensions in RDC. For example, the (*boot.dl_*?) file can be interpreted as *boot.dll*, which is

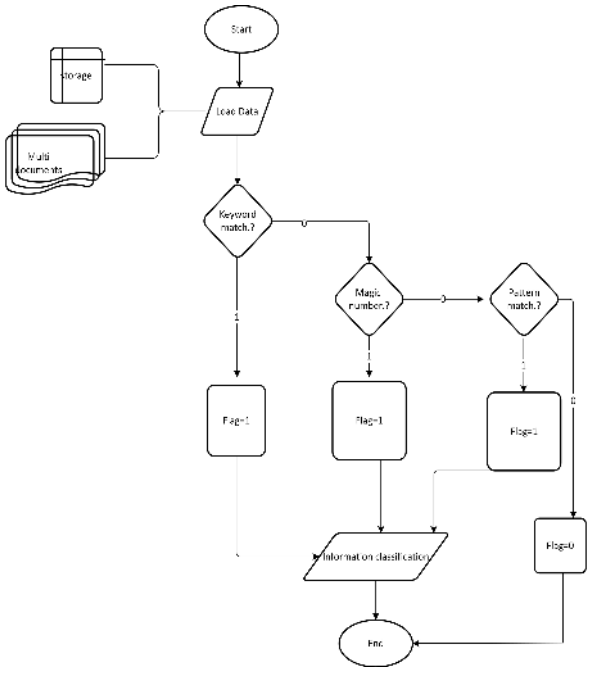


**FIGURE 3.** Proposed KMP classifier to detect suspicious files.

an essential file in loading the Windows operating system. "Fig. 5" shows the approach for identifying suspicious files in terms of extensions. The proposed methodology identified 1.2k files with altered and mismatched extensions.

### 3) PATTERN DESIGNED

In existing methods, a traditional technique like regex searches for pre-defined patterns such as credit card numbers, bank accounts, person names, zip codes, email addresses, and so on. The file will be labeled as suspicious if any of this information matches. The primary downside of the above approach is that they are static and cannot recognize dynamic patterns. These dynamic patterns include dictionary patterns and operator patterns like - emoticons(punctuations, letters, or numbers that represent pictorial icons); emoji's [44]; collocation passwords (sequence of words or terms that occur more often than would be anticipated by coincidence) [45]; and character substitutions(replacing characters with numbers or some special characters or converting them into an upper case). We propose several patterns for detecting user-defined or machine-generated passwords post-corpus analysis to address the shortcomings. For this, we gathered password breach incidents across the globe and created a password dictionary database for password verification. Furthermore, after carefully analyzing millions of passwords, we designed novel password patterns to determine new passwords that are unavailable in data breaches. A password dictionary is created using a global password breach containing over 50 million passwords from 67 leaked databases globally. The proposed patterns are stored in a centralized repository and compared to files in RDC. Files shall be flagged as relevant and sent for further evaluation upon matching as per the flow given

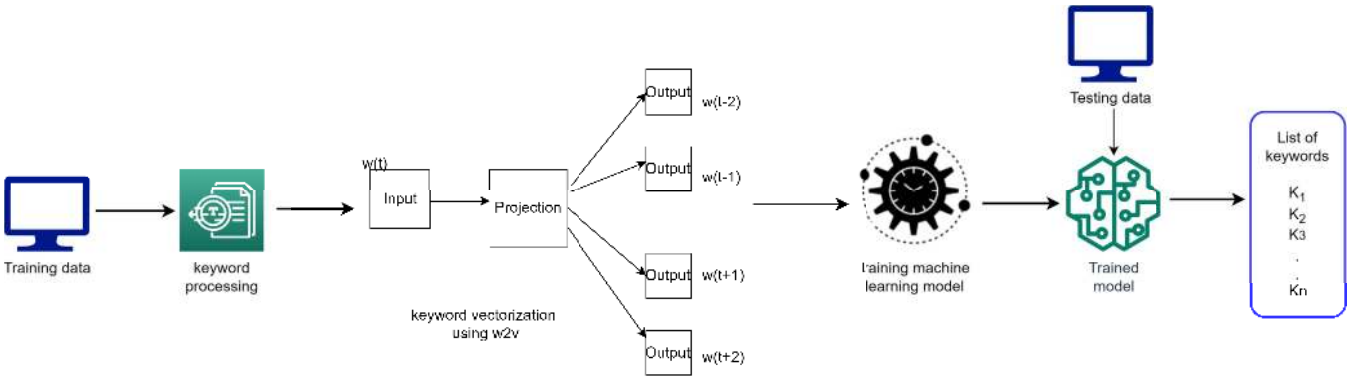


**FIGURE 4.** Flow of defining blacklisted keywords.

**Algorithm 2** Construction of Sensitive Keywords Using W2V and LDA

**Input:** $D_i$
**Output:** sensitive words
$bkw(T, V_i)$
**begin**
start
$V_c = random \;\; V_w = random$
**w2v**$(D_i, V_i)$
{
**for** *i in total iterations over T* **do**
**for** *center word in T* **do**
$\arg\max_\theta \left( \sum_{(w,c)\in D} \log \sigma\ (v_c \cdot v_w) + \sum_{(w,c)\in D'} \log \sigma\ (-v_c \cdot v_w) \right)$
**end**
**end**
}
**lda**$(V_i, K)$
{
$P(W, Z, \theta, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{j=1}^{M} P\left(\theta_j; \alpha\right) \prod_{i=1}^{K} P\left(\varphi_i; \beta\right) \prod_{t=1}^{N} P\left(Z_{j,t} \mid \theta_j\right) P\left(W_{j,t} \mid \varphi_{Z_{j,t}}\right)$
}
**end**
where $P(W, Z, \theta, \boldsymbol{\varphi}, \alpha, \beta)$ is total probability of LDA,
T is corpus, k is topics, j is document, W is word.

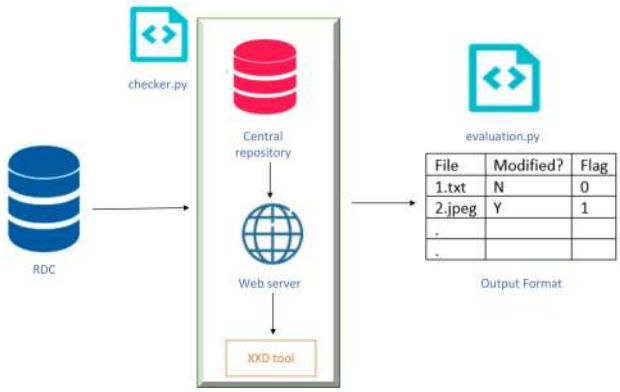


**FIGURE 5.** Identification of suspicious files based on magic number in KMPT.

in "Fig. 6." Furthermore, the file encoding type from the file metadata aids us in identifying sensitive files. Since providing all of the regex patterns is challenging, few pattern-matching expressions are available in "Fig. 6."

We simulated a training dataset incorporating most patterns to validate the accuracy and detection rate of traditional and proposed techniques. The comparison results are given in Table 2.

From Table 2, we can interpret that existing forensic pattern-matching algorithms and classic regex cannot identify math passwords, dictionary patterns, emoticons, emojis, and collocation passwords. In contrast, our proposed forensic pattern-matching algorithm could overcome all these downfalls and yield better results even though further improvement is needed. In the credit card category, 100% accuracy could not be achieved due to updated lengths in 2021 and 2022 master cards. As new email domains frequently emerge, a 100% email detection rate could not be achieved. Our proposed patterns identify emoticons and emojis with more than 92% accuracy. In the collocation passwords category, the existing approach failed to identify, whereas our approach identified passwords with 34.6% accuracy. So far, data cleaning, normalization, and preliminary suspicious data identification using the KMP classifier have been observed on RDC. After tier 2, the KMP classifier results in suspicious(label-1) and non-suspicious(label-0) files. The authors labeled the data into two categories based on this classification, and the dataset is prepared for automated machine learning classification. Since tier 3 includes a machine learning classifier, the KMP classifier's output is vectorized using the D2V model, and how the data is trained is given in the next section.

The complete pseudo code for tier 2 is given in the algorithm 3

**Algorithm 3** KMPT_classifier()

**Input:**processed data
**Output:**set of forensic relevant files
**begin**
n=fseek-1
$D_i \leftarrow$ read data
**def bkw**($D_i$)
{
dict(k,v) $\leftarrow D_i$
∀ bkdb ←dict(k,v)
**for** $i \leq n$ **do**

$$\prod \begin{cases} flag = 1 \text{ if } x \in \text{dict}(k, v) \\ flag = 0 \text{ if } x \notin \text{dict}(k, v) \end{cases}$$

**end**
}
**def** *file_ext*($D_i$)
{
load extensions database ext_db
$\sum_{i=0}^{n} ext\_db(k, v) = zip(ext, header)$
sus_files=[]
ext=[set of all extensions in RDC]
**for** *i =0 to n* **do**
extensions=D.rsplit('.',1)[-1]
**for** *i in extensions* **do**
**if** *i.read(24) =ext_db[i.values]* **then**
flag=0
**end**
flag=1
sus_files.append(i)
**end**
**end**
**for** *i in sus_files* **do**
**if** *(hex(i.read[24]==XXD file | head -n))* **then**
continue
**end**
**end**
}
**def patterns**($\mathbf{D}_i$)
{
Set directory name Set extension ext = doc| docx| txt | rtf | pdf | ppt | pptx | xls | xlsx | xps) | .odt | csv | xml **for** *i =0 to n* **do**
**for** *file in Data* **do**
**while** *file.rsplit('.',1)[−1] ∈ ext[]* **do**
**if** *tok_list[i].match($R_i$)* **then**
then sus_files.append(file)
**end**
continue
**end**
**end**
**end**
}
**end**

### D. TIER-3 DATA CLASSIFICATION: LINEAR SVM MODEL

This section explains how the KMPT result is vectorized and how the data is being trained with a machine-learning classifier. Since a machine learning text classifier works with supervised data, an unstructured corpus is now transformed into structured data with the help of the KMP classifier that uses D2V vectorization [46]. Though other models such as BOW [47], TF-IDF, W2V with skip-gram, Continous-BOW, and Distributed-BOW models, D2V is considered suitable in the current forensic analysis context because D2V adds one more vector in space. Furthermore, D2V can be used to acquire document similarities, label representations, and word embeddings. As a result, the D2V model with skip-gram is used for vectorization because skip-gram represents words better than the C-BOW model. Now, for a given context word $W_c$ and a given focus word $W_b$, the conditional probability is computed as shown in "(11)."

$$P(W_c)/P(W_b) = \frac{\exp\left(u_c^{C_o} v_b\right)}{\sum_i^{V_{OC}} \exp\left(u_i^{c_o} v_b\right)} \tag{11}$$

where $w_c$ is the probability of generating context word with c as the index, $W_b$ is the center word with b as the index, $u_c$ represents the context word and $v_b$ represents the center word, $C_o$ represents the corpus and $i$ represents the index, $V_{OC}$ represents the Vocabulary.

Assuming that context word $W_c$ is independently generated for any center word $V_b$ with window size=s, the probability of generating context words over given focus words on $V_{oc}$ is calculated using maximum likelihood function given in "(12)."

$$L(\theta) = \prod_{r=1}^{E_T} \prod_{-s \le p \le s, p \ne 0} P\left(\frac{w_{r+p}}{w_r}\right) \tag{12}$$

where $s$ is the window size, $p$ is the position of a word, $E_T$ is the total corpus and $\theta$ is the model parameter.

Since the skip-gram model parameters are $V_b$ and $u_c$ for each word in $V_{oc}$, model parameters $V_b$ and $u_c$ are learned and trained in the current context by maximizing the likelihood function given in "(13)."

$$-\sum_{r=1}^{E_T} \sum_{-s \le p \le s, p \ne 0} \log P\left(w^{(r+p)} \mid w^{(r)}\right) \tag{13}$$

We used Stochastic Gradient(SG) for updating model parameters to minimize the loss in "(13)." To determine *SG*,the log conditional probability for $V_b$ and $u_c$ should be calculated according to the "(14)."

$$\log P(w_c \mid w_b) = \mathbf{u}_c^{Co} \mathbf{V}_b - \log\left(\sum_{i \in \mathcal{V}_{\lfloor}} \exp\left(\mathbf{u}_i^{Co} \mathbf{v}_b\right)\right) \tag{14}$$

r1 =Λ(? =∗?[\D])(? = .∗?[\d])(? = .∗?[0 − 9])(? = .∗?[#?!@$%Λ& ∗ −]) ·{6, }$

r2=Λ(? = .∗[\d])(? = .∗[\D])(? = .∗[0 − 9])(? = .∗[!(A#\$% Λ&\∗])(? = .{8, })$

r3=Λ(? = .∗[\D\d])(? = .∗\d)(? = .∗[a$!%∗#?&])[\D\d\dA$!%∗#?&]{8, }$

r4=[\d] + [\D] + [0 − 9] + [S%&{[. . .] + [: punct :] + [Λ\w\s]+

r5= (? = · {9, })(? = .∗[\d])(? = .∗[\D])(? = .∗[0 − 9])(? = .∗\p{ Punct }).∗

r6 =(? = .{9, })(? = .∗?[Λ\w\s])(? = .∗?[0 − 9])(? = .∗?[\D])·∗?[\d]·∗

r7=/Λ(? = .∗[\d])(? = .∗[\D])(? = .∗[0 − 9])(? = .∗[SQ])(?!.∗[iIoO])\S{6, 12}$/

r8=(? = Λ · {8, }$) (? = .∗\d) (? = .∗ [!@#$%Λδ∗] +) (?![·\n])(? = .∗[\D]) (? = .∗[\d]) · ∗$

r9=/Λ(? =(? : [Λ\d]+ [\d]){2}) (? = (? : [Λ0 − 9]∗ [0 − 9]) {2}) (? = .∗[! − \/ : −@\[−\{− ∼]) ·{6, }$/i

**FIGURE 6.** Designed Patterns in KMPT.

**TABLE 2.** Efficiency of proposed patterns in detecting patterns.

| Pattern type | Example | Test data size | Detection rate in existing approach (%) | Detection rate of proposed approach (%) |
|---|---|---|---|---|
| Credit card | 11/13/14/15/16/19 digit | 1500 | 98.5 | 99.9 |
| Debit card | 13/16/19 digit | 1000 | 100 | 100 |
| SSID | 9 digits | 1000 | 100 | 100 |
| address | <name><d no><street> <locality> <zip><state> <country> | 1000 | 100 | 100 |
| Email | .com /. in/.edu/.gov/.net/ .it/ Hotmail/ .org/ .sg/ aol/ bol/ gmx/.ru | 1500 | 78.5 | 86.6 |
| Math passwords | 98*76/44-32+21% | 50 | <1 | 52.4 |
| Emoticons | :’( | :*) | 45 | <1 | 92% |
| Emoji’s | ` | 100 | <1 | 94.5 |
| Collocation passwords | Crystal clear/ heavy rain | 100 | <1 | 34.6 |
| Dictionary patterns | Key: value | 220 | <1 | 66.8 |
| Character substitution | cybersecurity-> $Yb3R3$cuR1TY | 50 | 0 | 22.1 |

Through differentiation,with respect to $V_b$ and all other word vectors, gradient can be obtained from “(15).”

$$\frac{\partial \log P\left(w_c \mid w_b\right)}{\partial \mathbf{V}_b} = \mathbf{u}_c - \sum_{p \in \mathcal{V}_\downarrow} P\left(w_p \mid w_b\right).\mathbf{u}_p \qquad (15)$$

To ensure that the data is relevant or irrelevant, supervised data classification algorithm is used in the next step. Text-supervised algorithms such as Multinomial Nave Bayes, Logistic Regression, SVM, and Random Forest are used in the machine learning world. When these models are evaluated, SVM with linear kernel delivers the best results for text classification, with improved accuracy, precision, recall, and F1-score. Table 8 compares different text classification models. All models are compared on a data set split 70:30 between training and testing data. Equation (16) is used in this work to polarise data into two major classes: suspicious and non-suspicious.

$$f(x) = sign(w^t x + b) \qquad (16)$$

where $b$ is a biased term for defining the boundary and $w$ as weight, in other words, hyperplane function $h(x)$ for linear separable classes is calculated by using “(17).”

$$h\left(x_i\right) = \begin{cases} +1 & if\ w.x\ +\ b \geq 0 \\ -1 & if\ w.x\ +\ b\ <\ 0 \end{cases} \qquad (17)$$

where $w$ is the vector and $x$ is the variable and $b$ is the biased term. It is also considered that functional margin is 1 for all support vectors as mentioned in “(17)” such that $r_i$ for all

suspicious files is one and $r_i$ for non-suspicious is -1 as given in "(18)."

$$r_i = \frac{y_i\left(w^t x + b\right)}{|w|} \geq \frac{1}{|w|} \tag{18}$$

Since "((16)),(17) and (18)" are helpful for linear separable classes, and as much of the files in RDC demand non-linear separable classes, these equations in turn, are used to classify the n-dimensional data into m-dimensional data where $n > m$

$$f\ (x) = sign\left(\sum_{i=1}^{N} a_i y_i K\ (x_i x) + b\right) \tag{19}$$

where $K(x_i, x_j)$ is $\varphi(x_i)^t \varphi\left(x_j\right)$ for linear kernel. Although RDC is multidimensional data with two classes (class 0 and class 1), the linear kernel produced the best results in this work when compared to other kernels. This work is evaluated using other kernels such as the Polynomial kernel ($P_k$), the Sigmoid kernel ($S_k$), and the Gaussian Radial basis Kernel ($RBF$), as shown below. This study also compared KMPT classification model with other kernels in terms of text classification, and the results are shown in section IV.

$$P_k \overset{K}{\leftarrow} \left(x_i, x_j\right) = a\left(x_i.x_j + b\right)^d \tag{20}$$

where $b$ is biased term, $a$ is constant, and $x_{i,j}$ are variables.

$$S_k \overset{K}{\leftarrow} \left(x_i, x_j\right) = \frac{1}{cosec\ (h)}\left(a\left(x_i.x_j\right) + b\right) \tag{21}$$

where $b$ is biased term, $a$ is constant, and $x_{i,j}$ are variables.

$$RBF \overset{K}{\leftarrow} \left(x_i, x_j\right) = \frac{exp\left(-\frac{1}{2}|x_i - x_j|^2\right)}{\sigma^2} \tag{22}$$

where $\sigma$ is variance and $|x_i - x_j|$ is Eucledian distance between variables

## IV. RESULTS AND DISCUSSION

The proposed methodology is applied to the T5 corpus, RDC, and documents related to security events crawled by a data crawler consisting of approximately 1.4 million documents. Table 3 presents a summary of the datasets gathered from various sources. This dataset is passed to the pre-processing engine, which tokenizes the documents, removes unnecessary common words, and identifies the root words. RDC contains over 5K orphaned documents, of which 28 percent of files are user related. Regex patterns are used in conjunction with a pre-processing engine to identify common patterns such as email, phone numbers, and SSIDs and remove noisy characters such as hyperlinks and tags. When Tier 1 techniques are applied to the corpus, 10,99,91,520 tokens are removed, thereby resulting in a 40% reduction rate in tokenization. Resultant files are then passed to the KMP engine, where 4554 sensitive keywords in 56,754 files are identified, 5,928 files are identified as suspicious by the metadata module, and 66,690 files are identified with suspicious patterns by the pattern module. A total of 1,29,372 files are detected as suspicious or relevant categories by the KMP engine

**TABLE 3. Unstructured Data set description.**

| | |
|---|---|
| Number of files | 14,82,000 |
| Number of sentences | 77,86,900 |
| Number of tokens | 18,33,19,200 |
| Number of unique tokens | 11,50,660 |
| Number of orphaned files | 5,928 |

"Fig. (7)" depicts the workflow of tier-1 and tier-2. The files extracted from the KMPT classifier are saved in a database and annotated with 0 and 1, with 0 indicating an irrelevant file and 1 indicating a relevant category. By revisiting the output of the KMP classifier, the authors went one step further in meticulously annotating the data. The data is vectorized using the D2V model rather than headlines alone in [35]. The authors used BOW, TF-IDF, and W2V embedding models with the KMPT classifier to vectorize the data. The D2V+KMP model produced the best results for Tier-2, as document-level embedding was observed in this study. After obtaining vectorized data from Tier-2, the data is fed into SVM for automated classification of suspicious or interesting files, as SVM is a good text classifier [48]. Different vectorization models are trained with the corpus and are tested with RDC to evaluate each model's classification metrics. In this scenario, each vectorized model is experimentally evaluated with all available SVM kernels and compared with our classification model, which is built upon D2V+linear SVM. In this work, the Linear kernel is denoted as L-SVM, the Polynomial kernel as P-SVM, Gaussian Radial Basis as G-SVM, and the Sigmoid kernel as S-SVM.

We compared our KMPT model vectorized with the BoW model with L-SVM, G-SVM, and S-SVM, which are vectorized using the BOW model. Table 4 compares the Bag-of-Words model-based KMPT with polynomial(P), Sigmoid(S), and Gaussian(G) SVM kernels. While comparing KMP with other classifiers, this work provides True Positive (TP) and True Negative (TN) concerning Precision, Recall, F1-score, Accuracy, specificity, Hamming Loss, and MCC. It can be understood that the proposed KMPT with D2V vectorization model outperforms stand-alone models such as Tf-IDF, W2V, and BoW. Table 5 provides a detailed comparison of the proposed methodology with L-SVM and other SVM kernels using word-level vectorization. Table 6 compares KMPT to other SVM kernels using the TF-IDF vectorized method. Table 7 compares KMPT to other SVM kernels by using vectorization at the document level. It is self-evident that compared to other SVM kernel classifications, KMPT and linear SVM kernel yielded the best results throughout all the vectorization models.

### A. PERFORMANCE METRICS IN EVALUATING KMPT

True Positive: The file is correctly classified as a relevant or sensitive category and denoted with binary 1.

True Negative: The file is correctly classified as an irrelevant or non-sensitive category and denoted with binary 0.

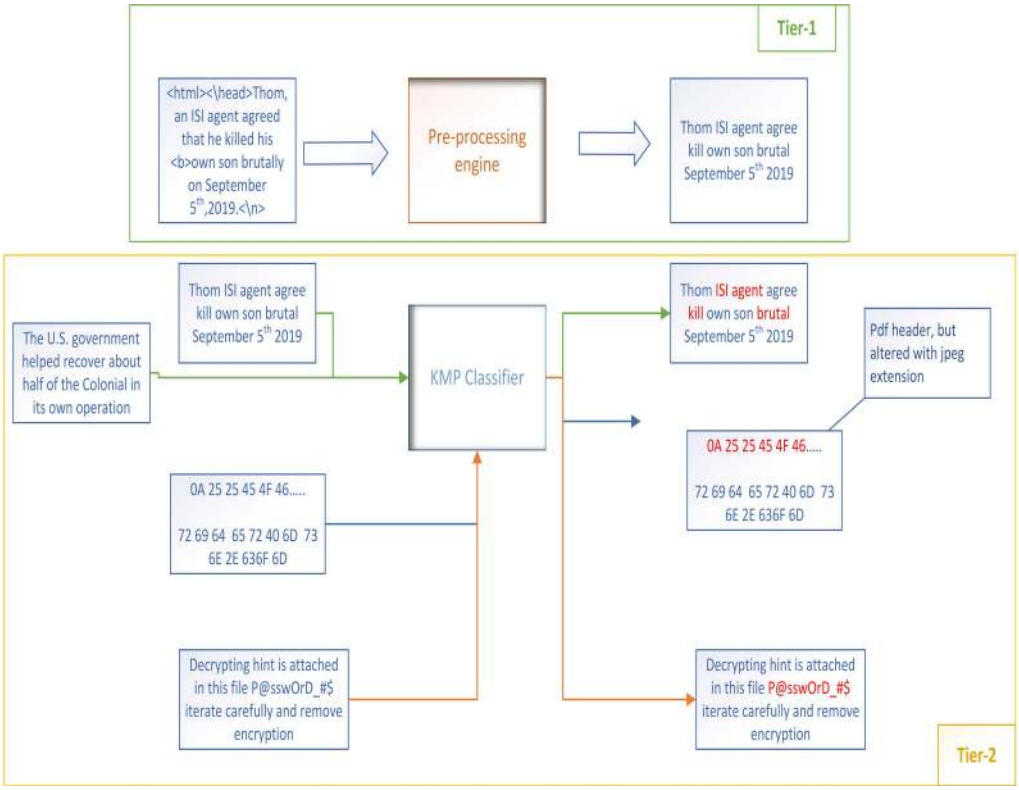


**FIGURE 7.** Experimental evaluation of KMPT Tier-1 and Tier-2.

False positive: The file is falsely classified as a relevant category, even though it is an irrelevant category by nature.

False Negative: The file is falsely classified into an irrelevant category, even though it is a relevant category by nature.

Recall or sensitivity: It is a metric that states how many are classified in the overall actual positive class. Higher recall value reveals that data is highly predicted and better performance of the model.

$$Recall = \frac{TP}{TP + FN}$$

Precision: It is a metric used in this work to identify how many class-1 files are identified in the overall class-1 category.

$$Precision = \frac{TP}{TP + FP}$$

Specificity: It evaluates a model's potential to predict class-1 files of each available class.

$$Specificity = \frac{TN}{(TN + FP)}$$

Accuracy: This is the base metric used to evaluate our model by identifying precise classification classes over total classification classes.

$$Accuracy = \frac{TP + TN}{(TP + TN + FP + FN)}$$

F1–Score: F1-Score is the weighted average of Precision and Recall. F1–Score is an instrumental performance measurement technique. It is widely used in scenarios when the model produces high recall and low precision or low recall and high precision. In such scenarios, measuring the model's performance is very complicated. F1-score makes Precision and Recall comparable. It uses the harmonic mean instead of the arithmetic mean.

$$F1 - score = 2 * \frac{Recall * Precision}{(Recall + Precision)}$$

ROC curve: A receiver operating characteristic curve is a graph that illustrates a classification model's performance across all classification levels by considering a True positive rate and a False positive rate. It evaluates a classifier's capacity to distinguish among each class in a balanced classification.

$$ROC = \frac{1}{2}\left\{\frac{TP}{(TP + FN)}\frac{TN}{(TN + FP)}\right\}$$

Matthew's correlation coefficient: This statistical metric evaluates the correlation between predicted values and actual values that ranges from -1 to 1, being a good classifier towards 1.

$$MCC = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP + FN)(TP + FP)(TN + FP)(TN + FN)}}$$

**TABLE 4.** Performance evaluation of BoW based KMPT with SVM.

| X_test=0.30 | Precision | | Recall | | F1- Score | | Support | | Specificity | MCC | Hamming loss |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | TP | TN | TP | TN | TP | TN | TP | TN | | | |
| BoW+P-SVM | 0.54 | 0.98 | 1.0 | 0.21 | 0.70 | 0.34 | 214 | 231 | 0.92 | 0.35 | 0.38 |
| BoW+G-SVM | 0.77 | 0.92 | 0.93 | 0.74 | 0.85 | 0.82 | 214 | 231 | 0.90 | 0.67 | 0.16 |
| BoW+S-SVM | 0.76 | 0.95 | 0.96 | 0.71 | 0.85 | 0.82 | 214 | 231 | 0.86 | 0.69 | 0.16 |
| BOW+KMPT | 0.80 | 0.87 | 0.87 | 0.80 | 0.85 | 0.83 | 214 | 231 | 0.92 | 0.67 | 0.15 |

**TABLE 5.** Performance evaluation of W2V based KMPT with SVM.

| X_test=0.30 | Precision | | Recall | | F1-Score | | Support | | Specificity | MCC | Hamming loss |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | TP | TN | TP | TN | TP | TN | TP | TN | | | |
| W2V+G-SVM | 0.59 | 0.60 | 0.63 | 0.61 | 0.61 | 0.57 | 223 | 222 | 0.83 | 0.63 | 0.18 |
| W2V+P-SVM | 0.52 | 0.56 | 0.76 | 0.31 | 0.62 | 0.40 | 222 | 222 | 0.86 | 0.56 | 0.22 |
| W2V+S-SVM | 0.58 | 0.59 | 0.61 | 0.55 | 0.59 | 0.57 | 223 | 222 | 0.82 | 0.64 | 0.17 |
| W2V+KMPT | 0.60 | 0.62 | 0.79 | 0.65 | 0.63 | 0.59 | 213 | 222 | 0.86 | 0.64 | 0.17 |

**TABLE 6.** Performance evaluation of TF-IDF based KMPT with SVM.

| X_test=0.30 | Precision | | Recall | | F1-Score | | Support | | Specificity | MCC | Hamming loss |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | TP | TN | TP | TN | TP | TN | TP | TN | | | |
| TI+S-SVM | 0.80 | 0.80 | 0.79 | 0.81 | 0.80 | 0.80 | 221 | 224 | 0.84 | 0.65 | 0.17 |
| TI+P-SVM | 0.80 | 0.53 | 0.20 | 0.95 | 0.32 | 0.68 | 229 | 216 | 0.91 | 0.42 | 0.31 |
| TI+G-SVM | 0.84 | 0.86 | 0.86 | 0.84 | 0.85 | 0.85 | 220 | 225 | 0.85 | 0.65 | 0.17 |
| TI+KMPT | 0.85 | 0.87 | 0.86 | 0.84 | 0.86 | 0.87 | 227 | 218 | 0.89 | 0.67 | 0.16 |

**TABLE 7.** Performance evaluation of D2V based KMPT with SVM.

| X_test=0.30 | Precision | | Recall | | F1-Score | | Support | | Specificity | MCC | Hamming loss |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | TP | TN | TP | TN | TP | TN | TP | TN | | | |
| D2V+P-SVM | 0.72 | 0.86 | 0.87 | 0.69 | 0.79 | 0.77 | 211 | 234 | 0.88 | 0.65 | 0.17 |
| D2V+G-SVM | 0.82 | 0.87 | 0.87 | 0.82 | 0.84 | 0.85 | 211 | 234 | 0.88 | 0.74 | 0.13 |
| D2V+S-SVM | 0.83 | 0.86 | 0.85 | 0.84 | 0.84 | 0.85 | 211 | 234 | 0.87 | 0.73 | 0.13 |
| D2V+KMPT | 0.91 | 0.86 | 0.95 | 0.87 | 0.92 | 0.93 | 211 | 234 | 0.93 | 0.75 | 0.11 |

Hamming loss: Hamming loss (HL) metric is the fraction of misclassified entities ranging from 0(best classification) to 1(worst classification).

$$\text{Hamming Loss } = \frac{1}{nL}\sum_{x=1}^{L}\sum_{y=1}^{N} Q_{x,y} \oplus P_{x,y}$$

where $n$ is training example, $Q_{x,y}$ and $P_{x,y}$ are boolean $i^{th}$ predictions containing $y^{th}$ label.

In Table 4, we experimentally evaluated our KMPT model based on the BOW vectorization model and compared our model with different SVM kernels based on the BoW model. Even though P-SVM identified class 1 entities with 98% accuracy, it identified class 0 with 54% accuracy. Furthermore, the same model resulted in a 100% sensitivity rate in detecting class-0, yet it could only identify class-1 with 21%. The results show that our proposed model yielded a balanced classification of class-0 and class-1 in terms of accuracy, sensitivity, and F1 score. As accuracy does not consider class imbalance, considering accuracy alone will be misleading. Therefore, we consider F1-score and ROC curves as the primary metric to evaluate our model.

In Table 5, we experimentally evaluated our KMPT model based on the W2V vectorization model and compared our model with different SVM kernels based on the W2V model. From the above results, even though W2V-based classification led to poor classification results, it is evident that our proposed model based on w2v gave good classification results in terms of accuracy, sensitivity, and F1-score.

In Table 6, we experimentally evaluated our KMPT model based on Tf and IDF and compared our model with different SVM kernels based on the TF-IDF model. Our model resulted in 85% precision, 86% recall, and 86% F1-score, which is better than other models.

In Table 7, we experimentally evaluated our KMPT model based on the D2V model and compared our model with different SVM kernels based on the D2V model. Compared to previous vectorization models like BoW, TF-IDF, and W2V, our model with D2V yielded the best results with 91% precision, 95% recall, and 92% F1-score. Since the corpus is imbalanced, F1-score is an accurate metric rather than accuracy to check the model consistency. Finally, the overall proposed system is evaluated with various *feature_size*$(X)$ where $X = 40, 30, 20$, and 10, and performance metrics such as precision, recall, f1-score, and accuracy are calculated and shown in "Fig. 8" for feature size $f(x) = 0.30$. When comparing the overall system to different features, the performance metrics for *feature_size*$(X = 30)$ yielded better results.

**TABLE 8.** Performance metrics of KMPT-SVM based on different vectorization models for $f(x) = 0.30$.

| KMP with existing work | Precision | Recall | F1-Score | Accuracy |
|---|---|---|---|---|
| BoW + SVM | 0.77 | 0.93 | 0.85 | 0.85 |
| Tf-Idf + SVM | 0.85 | 0.86 | 0.86 | 0.85 |
| W2V + SVM | 0.82 | 0.87 | 0.84 | 0.84 |
| D2V + SVM | 0.60 | 0.65 | 0.63 | 0.61 |
| KMPT | 0.91 | 0.95 | 0.92 | 0.90 |

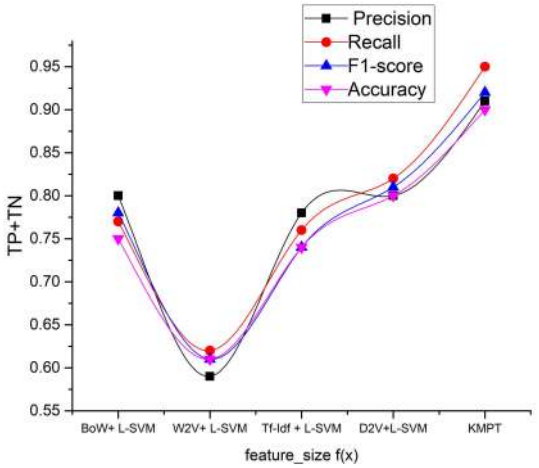


**FIGURE 8.** Performance evaluation of KMPT based on vector models.

In Table 8, BOW+SVM represents that the tier 1 result is vectorized using the BOW model and labeled according to DHS keywords and existing regex patterns. The resultant data is trained with L-SVM for automated classification resulting in 74% accuracy. Similarly, TF-IDF+SVM, W2V+SVM, and D2V+SVM represent that tier 1 result is vectorized according to each model and labeled according to existing DHS keywords and regex patterns. The resultant data is trained with linear SVM for data classification resulting in 74%, 61%, and 80% accuracy. KMPT represents the result of the overall three-tier proposed system and thus yields best precision, recall, f1-score, and accuracy.

"Fig. 9" represents the resultant ROC curve for the proposed methodology and "Fig. 10" represents confusion matrix and it is evident that TP and TN outperforms FP and FN resulting in best classification.

### B. EVALUATION OF TIME COMPLEXITY

It is the computational complexity often approximated by counting the number of elementary operations executed by the algorithm. In tier 1, the whole corpus is matched against the SHA-256 hash corpus that contains nearly 10 million. For such an enormous hash corpus, as sorting is recommended, we, therefore, sorted our corpus, which took $O(nlog(n))$ in terms of time complexity that is better than $O(n^2)$ in generic hash unsorted search. For tier 2,

Where n is number of examples, D is dimensions of data(640) and V is the size of vocabulary(1 million). Training

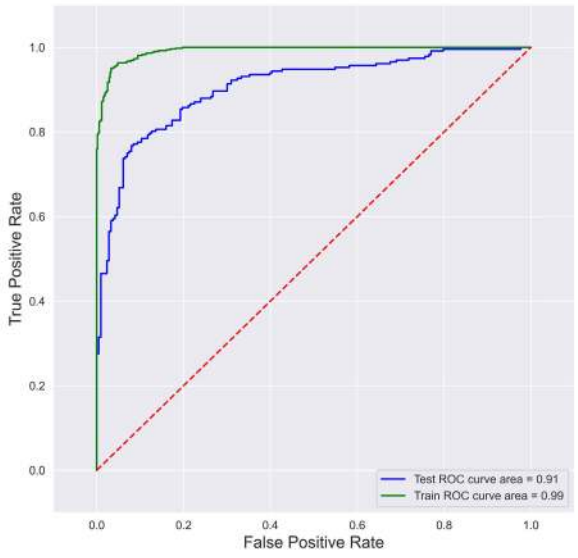


**FIGURE 9.** ROC curve for the proposed system.

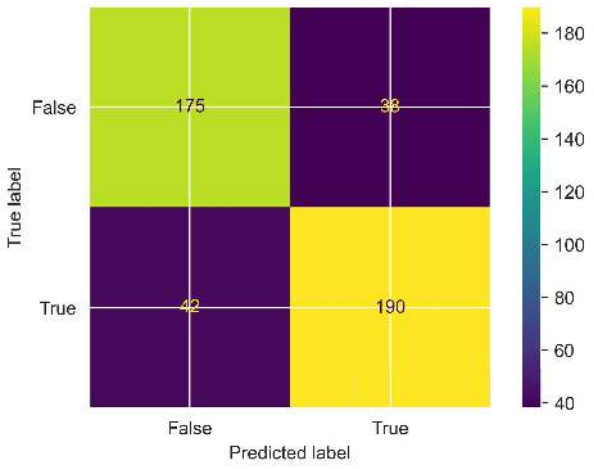


**FIGURE 10.** Confusion Matrix for the proposed system.

```
Word2Vec(corpus, size=4000, window=30, min_count
    =1, workers=50, iter=100, alpha=0.0001) and
    lda_model = LDA(corpus=doc_term_matrix,
    id2word=dictionary, num_topics=100,
    random_state=20, chunksize=1000, passes=100)
```

**Listing 2.** Custom stop words module.

complexity for W2V is proportional to $O = E \times T \times Q$, where $E$ is epoch size( 5 in this case), $T$ is words in training dataset (1 million in this case), and $Q$ is $N \times D + D \times log_2 V$,

Therefore, by considering hierarchical SoftMax (hs)=1, words are differently encoded and increase as V increases; this results in the number of training computations by log(V), thereby yielding in O(n log (V)) time complexity. Training complexity for LDA is *O(DxLxT)*, (D- documents, T-topics, L -unique words in D) and proportional to *n_samples * the number of iterations*. Therefore the overall complexity results

in *O(n log N)* as each operation in the input data has logarithmic time complexity. The running time or predicting time for linear SVM is *O(k*d)*, where *k* is the support vector/vectors and *d* is the total number of data points.

## V. CONCLUSION

The proposed three-tier framework in this work reduced forensic investigation delay by avoiding the evaluation of unwanted files in massive data. This system also solved the problems associated with two-tier models, such as identifying and labeling the relevant forensic files. In Tier-1, due to Inverse document frequency, the custom stop words module identified an additional 248 stop words in the corpus resulting in 4.1% elimination of tokens in RDC, and the forensic classification process was accelerated. In Tier-2, w2v with the lda model detected 3,709 novel sensitive keywords in RDC and 845 from DHS, totaling 4554 blacklisted keywords. As a result of our model, 11.48% of files are identified as suspicious in RDC. The three-stage metadata extension module identified 970 file extensions in addition to the 25K extensions, classified 1.81% of files in RDC as suspicious. We created 280 unique patterns in addition to the existing patterns to identify the dynamic patterns in RDC. Our unique pattern module identified 34.6% collocation passwords, 92% emoticons, 94.5% emojis, 22% character substitution passwords, 66.8% dictionary patterns, and 52.4% math passwords, apart from 461 user passwords and 268 machine-generated passwords. Due to this, RDC's level of suspicious files is extended to 16.06%. From tier 2, data was labeled as per the KMP classifier, and the linear SVM model was trained on the KMPT classifier and classified the forensic relevant data with 91% accuracy, 91% precision, 95% recall and 92% f1-score with 0.75% MCC relevancy and 0.11% of hamming loss with quasi-linear complexity.

## REFERENCES

[1] N. L. Beebe and J. G. Clark, "A hierarchical, objectives-based framework for the digital investigations process," *Digit. Invest.*, vol. 2, no. 2, pp. 147–167, Jun. 2005.

[2] *Crime in India 2020 National Crime Records Bureau*, Ministry Home Affairs, India, 2022.

[3] A. Guarino, "Digital forensics as a big data challenge," in *Proc. ISSE Securing Electron. Bus. Processes*. Wiesbaden, Germany: Springer, pp. 197–203, 2013.

[4] B. Hitchcock, N.-A. Le-Khac, and M. Scanlon, "Tiered forensic methodology model for digital field triage by non-digital evidence specialists," *Digit. Invest.*, vol. 16, pp. S75–S85, Mar. 2016.

[5] NIST. (2022). *Current RDS Hash Sets*. [Online]. Available: https://www.nist.gov/itl/ssd/software-quality-group/national-software-reference-library-nsrl/nsrl-download/current-rds

[6] P. Joseph and J. Norman, "Forensic corpus data reduction techniques for faster analysis by eliminating tedious files," *Inf. Secur. J. A, Global Perspective*, vol. 28, nos. 4–5, pp. 136–147, 2019.

[7] G. Leurent and T. Peyrin, "SHA-1 is a shambles: First chosen-prefix collision on SHA-1 and application to the PGP web of trust," in *Proc. 29th USENIX Secur. Symp., (USENIX Security)*. Berkeley, CA, USA: USENIX Association, Aug. 2020, pp. 1839–1856.

[8] Z. E. Rasjid, B. Soewito, G. Witjaksono, and E. Abdurachman, "A review of collisions in cryptographic hash function used in digital forensic tools," *Proc. Comput. Sci.*, vol. 116, pp. 381–392, 2017.

[9] J. Garcia, "An evaluation of side-information assisted forensic hash matching," in *Proc. IEEE 38th Int. Comput. Softw. Appl. Conf. Workshops*, Jul. 2014, pp. 331–336.

[10] N. C. Rowe, "Identifying forensically uninteresting files using a large corpus," in *Proc. Int. Conf. Digit. Forensics Cyber Crime*. Cham, Switzerland: Springer, pp. 86–101, 2013.

[11] L. Fang, T. Wu, Y. Qi, Y. Shen, P. Zhang, M. Lin, and X. Dong, "Improved collision detection of MD5 with additional sufficient conditions," *Electron. Res. Arch.*, vol. 30, no. 6, pp. 2018–2032, 2022.

[12] G. Kessler, "File signatures table," 2012.

[13] W. Jo, Y. Shin, H. Kim, D. Yoo, D. Kim, C. Kang, J. Jin, J. Oh, B. Na, and T. Shon, "Digital forensic practices and methodologies for AI speaker ecosystems," *Digit. Invest.*, vol. 29, pp. S80–S93, Jul. 2019.

[14] X. Du and M. Scanlon, "Methodology for the automated metadata-based classification of incriminating digital forensic artefacts," in *Proc. 14th Int. Conf. Availability, Rel. Secur.*, Aug. 2019, pp. 1–8.

[15] M. A. Wani, A. AlZahrani, and W. A. Bhat, "File system anti-forensics—Types, techniques and tools," *Comput. Fraud Secur.*, vol. 2020, no. 3, pp. 14–19, Jan. 2020.

[16] A. Scholey and P. B. Zadeh, "A digital forensics live suspicious activity toolkit to assist investigators with sexual harm prevention order monitoring," in *Proc. IEEE Conf. Dependable Secure Comput. (DSC)*, Jun. 2022, pp. 1–6.

[17] G. Horsman, "Technical reporting in digital forensics," *J. Forensic Sci.*, vol. 67, no. 6, pp. 2458–2468, Nov. 2022.

[18] H. Arshad, A. Jantan, and E. Omolara, "Evidence collection and forensics on social networks: Research challenges and directions," *Digit. Invest.*, vol. 28, pp. 126–138, Mar. 2019.

[19] H. Kwon, S. Lee, and D. Jeong, "User profiling via application usage pattern on digital devices for digital forensics," *Exp. Syst. Appl.*, vol. 168, Apr. 2021, Art. no. 114488.

[20] L. Peng, X. Zhu, and P. Zhang, "A machine learning-based framework for mobile forensics," in *Proc. IEEE 20th Int. Conf. Commun. Technol. (ICCT)*, Oct. 2020, pp. 1551–1555.

[21] D. E. Salhi, A. Tari, and M. T. Kechadi, "Using clustering for forensics analysis on Internet of Things," *Int. J. Softw. Sci. Comput. Intell.*, vol. 13, no. 1, pp. 56–71, Jan. 2021.

[22] U. Noor, Z. Anwar, T. Amjad, and K.-K.-R. Choo, "A machine learning-based FinTech cyber threat attribution framework using high-level indicators of compromise," *Future Gener. Comput. Syst.*, vol. 96, pp. 227–242, Jul. 2019.

[23] I. Vayansky and S. A. P. Kumar, "A review of topic modeling methods," *Inf. Syst.*, vol. 94, Dec. 2020, Art. no. 101582.

[24] D. Sun, X. Zhang, K.-K.-R. Choo, L. Hu, and F. Wang, "NLP-based digital forensic investigation platform for online communications," *Comput. Secur.*, vol. 104, May 2021, Art. no. 102210.

[25] A. Agrawal, W. Fu, and T. Menzies, "What is wrong with topic modeling? And how to fix it using search-based software engineering," *Inf. Softw. Technol.*, vol. 98, pp. 74–88, Jun. 2018.

[26] X. Luo, "Efficient English text classification using selected machine learning techniques," *Alexandria Eng. J.*, vol. 60, no. 3, pp. 3401–3409, Jun. 2021.

[27] M. Hina, M. Ali, A. R. Javed, G. Srivastava, T. R. Gadekallu, and Z. Jalil, "Email classification and forensics analysis using machine learning," in *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI)*, Oct. 2021, pp. 630–635.

[28] A. M. Qadir and A. Varol, "The role of machine learning in digital forensics," in *Proc. 8th Int. Symp. Digit. Forensics Secur. (ISDFS)*, Jun. 2020, pp. 1–5.

[29] A. Kumar, V. Singh, T. Ali, S. Pal, and J. Singh, "Empirical evaluation of shallow and deep classifiers for rumor detection," in *Proc. Adv. Comput. Intell. Syst.* Singapore: Springer, pp. 239–252, 2020.

[30] F. Iqbal, M. Debbabi, and B. Fung, "Artificial intelligence and digital forensics," in *Machine Learning for Authorship Attribution and Cyber Forensics*. Cham, Switzerland: Springer, 2020, pp. 139–150.

[31] L. Bozarth and C. Budak, "Keyword expansion techniques for mining social movement data on social media," *EPJ Data Sci.*, vol. 11, no. 1, p. 30, May 2022.

[32] Y. Guo, J. Liu, W. Tang, and C. Huang, "Exsense: Extract sensitive information from unstructured data," *Comput. Secur.*, vol. 102, Mar. 2021, Art. no. 102156.

[33] L. Ignaczak, G. Goldschmidt, C. A. D. Costa, and R. D. R. Righi, "Text mining in cybersecurity: A systematic literature review," *ACM Comput. Surveys*, vol. 54, no. 7, pp. 1–36, Sep. 2022.

[34] F. B. Rodrigues, W. F. Giozza, R. de Oliveira Albuquerque, and L. J. G. Villalba, "Natural language processing applied to forensics information extraction with transformers and graph visualization," *IEEE Trans. Computat. Social Syst.*, early access, Apr. 5, 2022, doi: 10.1109/TCSS.2022.3159677.
[35] H. Jo, J. Kim, P. Porras, V. Yegneswaran, and S. Shin, "GapFinder: Finding inconsistency of security information from unstructured text," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 86–99, 2021.
[36] D. Gary Miner, J. Elder, and A. R. Nisbet, *Practical Text Mining and Statistical Analysis for Non-Structured Text Data Applications*. Cambridge, U.K.: Academic, 2012.
[37] M. Anandarajan, C. Hill, T. Nolan, F. Dai, and R. Maitra, "Practical text analytics: Maximizing the value of text data," *Technometrics*, vol. 62, pp. 1–286, Apr. 2020, doi: 10.1080/00401706.2020.1744910.
[38] C. Paulsen and R. Byers, *Glossary of Key Information Security Terms*. Gaithersburg, MD, USA: National Institute of Standards and Technology, Jul. 2019.
[39] X. Wei, C. Zhang, X. Yu, and Z. Zhuo, "A feature extracting and matching system based on magic-number and AC-algorithm," *Commun. Comput. Inf. Sci.*, vol. 1424, pp. 140–151, Jun. 2021.
[40] M. Paolanti and E. Frontoni, "Multidisciplinary pattern recognition applications: A review," *Comput. Sci. Rev.*, vol. 37, Aug. 2020, Art. no. 100276.
[41] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt, "Bringing science to digital forensics with standardized forensic corpora," *Digit. Invest.*, vol. 6, pp. S2–S11, Sep. 2009.
[42] G. E. Pibiri and R. Venturini, "Techniques for inverted index compression," *ACM Comput. Surveys*, vol. 53, no. 6, pp. 1–36, Dec. 2020.
[43] F. Naït-Abdesselam, A. Darwaish, and C. Titouna, "Malware forensics: Legacy solutions, recent advances, and future challenges," in *Advances in Computing, Informatics, Networking and Cybersecurity*. Cham, Switzerland: Springer, 2022, pp. 685–710.
[44] Q. Bai, Q. Dan, Z. Mu, and M. Yang, "A systematic review of emoji: Current research and future perspectives," *Frontiers Psychol.*, vol. 10, p. 2221, Oct. 2019.
[45] B. Liu and J. Wang, "Collocation features in translated texts based on English analogy corpus," *Sci. Program.*, vol. 2022, pp. 1–7, Mar. 2022.
[46] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. 31st Int. Conf. Mach. Learn.*, E. P. Xing and T. Jebara, Eds. Beijing, China, vol. 32, Jan. 2014, pp. 1188–1196.
[47] R. Zhao and K. Mao, "Fuzzy bag-of-words model for document representation," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 2, pp. 794–804, Apr. 2018.
[48] D. Rogers, A. Preece, M. Innes, and I. Spasić, "Real-time text classification of user-generated content on social media: Systematic review," *IEEE Trans. Computat. Social Syst.*, vol. 9, no. 4, pp. 1154–1166, Aug. 2022.

**D. PAUL JOSEPH** (Graduate Student Member, IEEE) received the Bachelor of Technology and Master of Technology degrees in computer science and engineering, in 2012 and 2015, respectively. He is currently a Ph.D. Scholar with the Vellore Institute of Technology. With hands on experience in ethical hacking, he joined as a full-time Ph.D. Scholar at VIT in the area of cyber security and digital forensics. His research interests include cyber security, digital forensics, NLP, ML, and information security. He received the Prestigious Fellowship under UGC, India, and designated as a Junior Research Fellow. He is also a Senior Research Fellow and carrying his research in digital forensics. He is also a Graduate Member of ACM. He received the Best Researcher Award, in 2017.

**P. VISWANATHAN** (Senior Member, IEEE) received the Diploma degree in computer technology from the Sreenivasa Polytechnic College, State Board of Technical Education, India, in 1998, the Bachelor of Engineering degree in computer science engineering from Madurai Kamaraj University, Madurai, India, in 2002, the Master of Engineering degree in computer science engineering from Annamalai University, Chidambaram, India, in 2006, and the Doctorate of Philosophy degree from the School of Computer Science Engineering, Vellore Institute of Technology, India, in 2014. He is currently a Professor at the School of Computer Science and Engineering, Vellore Institute of Technology. His current research interests include digital image processing, machine learning, cloud computing, the Internet of Things and computer graphics in shape analysis, description, and segmentation. He was a Senior Member of ACM and was a recipient of several academic and research awards, Best Poster Award from Indian Science Congress, in 2007, and VIT Most Active Researcher Award (2010–2021).