Received May 18, 2018, accepted June 23, 2018, date of publication June 28, 2018, date of current version July 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2851228

A Novel Mechanism for Fast Detection of Transformed Data Leakage

XIAOHONG HUANG[1], YUNLONG LU[1], DANDAN LI[1], AND MAODE MA[2]
[1]Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798

Corresponding author: Yunlong Lu (luyunlong@bupt.edu.cn)

This work was supported by the Joint Funds of the National Natural Science Foundation of China and Xinjiang under Project U1603261.

ABSTRACT Data leakage is a growing insider threat in information security among organizations and individuals. A series of methods has been developed to address the problem of data leakage prevention (DLP). However, large amounts of unstructured data need to be tested in the big data era. As the volume of data grows dramatically and the forms of data become much complicated, it is a new challenge for DLP to deal with large amounts of transformed data. We propose an adaptive weighted graph walk model to solve this problem by mapping it to the dimension of weighted graphs. Our approach solves this problem in three steps. First, the adaptive weighted graphs are built to quantify the sensitivity of the tested data based on its context. Then, the improved label propagation is used to enhance the scalability for fresh data. Finally, a low-complexity score walk algorithm is proposed to determine the ultimate sensitivity. Experimental results show that the proposed method can detect leaks of transformed or fresh data fast and efficiently.

INDEX TERMS Data leaks, weighted graphs, data transformation, label propagation, score walk.

I. INTRODUCTION

Data leakage (or data loss) is a term used in the information security field to describe unwanted disclosures of information [1]. Unlike the traditional security threats from the outside, data leakage is mainly caused by the insider who may leak data to the outside unauthorized entities. Thus, traditional security measures (i.e. firewalls, intrusion detection systems, anti-virus) are no longer valid due to lack of understanding of data semantics. However, data leakage incidents happen from time to time, bringing serious damage continuously. In November 2010, WikiLeaks leaked over 251,287 U.S. diplomatic cables [2], revealing the largest number of classified documents at that time. In June 2013, Edward Snowden leaked the senior confidential documents of the US National Security Agency (NSA), which exposed US spy program [3]. In October 2017, Yahoo reported that 3 billion accounts had been breached [4], covering every Yahoo account at the time. The security of data has attracted wide attention in many fields, for example, the recently developed Internet of Things [5]. According to Kaspersky Lab IT Security Risks Report 2016 [6], data protection is the main area of concern, with 80% of businesses saying it is their major concern. 43% of companies have experienced data loss due to the carelessness of employees.

To address the problem of data leakage detection (DLD), plenty of research work has been done with the use of hash fingerprinting, n-gram [7], statistical methods [8], [9] and so on. With the rapid development of Internet, many new communication technologies (e.g., Device-to-Device technology [10]) have emerged. As a result, the volume of data grows dramatically and the forms of data becomes much complicated. This brings new challenge to DLD. Therefore, new DLD method is required with better tolerance of data transformation and higher efficiency to deal with the large amounts of unstructured data in long patterns. Considering a common example scenario: an employee of an organization tries to leak a sensitive file to unauthorized outsiders, and he may deliberately make some transformation to the file (e.g, add some non-sensitive content to the original file, modify or delete some content) to escape detection. Existing methods mainly rely on the appearance of sensitive keywords and ignore the context, which may cause: a sensitive document may be detected as nonsensitive because the file is transformed through modifying some contents, resulting in low detection accuracy towards the transformed data. Some work has been done to detect the transformed data [7], [11], [12], however, most of them only consider simple cases, such as adding or deleting some content from the original file.

The data transformation in real application scenarios is much complex. As data transformation is quite common in real situation, a large number of sensitive data may be leaked once the detection can't tolerate the transformed data. Hence, how to better tolerate the data transformation is of vital importance for DLD.

In this paper, a novel model named Adaptive weighted Graph Walk model (AGW) is proposed to detect transformed data. In this model, all the documents are represented by graphs. The sensitive context weights in the form of node weights and edge weights are defined in the graph to improve the detection accuracy towards the transformed data. The context weight is able to quantify the sensitivity of the keywords adaptively based on the context around the keywords. The proposed solution aims to detect large amounts of newly generated, extensively transformed data accurately and efficiently. The main contributions are as follows.

- To better tolerate the long transformed data, we define an adaptive context weight mechanism to quantify the sensitivity of the keyword based on its context. The complex documents are further represented by weighted context graphs, containing both key terms and contextual information.
- To make up for the limitation of the template and increase the scalability, we also take the data semantics into consideration. An improved label propagation algorithm (LPA) is used to tag the same label on the highly relevant terms of the tested graphs. We also merge the context graphs of each file into a general one - the template graph, to preserve more correlation information between key terms and enhance the overall sensitive context.
- To deal with the large amounts of data, we propose an algorithm with low-complexity. With a weight reward and penalty mechanism, the algorithm quantifies the sensitivity of the tested documents by one walk on their graphs, which allows the detection to be implemented in real time.

The remaining of this paper is organized as follows. In Section II, related work about data leak detection is summarized. In Section III, the proposed model is presented and the related algorithms is discussed in detail. The performance evaluation is shown in Section IV. Finally, Section V summarizes the paper.

II. RELATED WORK

In this section, related research work about data leakage detection/prevention is summarized. In view of the serious damage of data leakage, the research on data leakage detection is far from sufficient and is relatively new. Researchers have paid more and more attention on data leak detection recently. Existing research can mainly be divided into two categories: content analysis and context analysis methods.

A. CONTENT ANALYSIS

The content analysis methods include rule-based methods, fingerprinting related methods and statistical methods. Among the rule-based methods, predefined regular expression is widely used in DLD. It scans the tested text with predefined rules. The sensitivity of the text depends on the extent to which it matches the regular expression. Rule-based methods can detect the exact data leakage rapidly, but they will fail to detect the new data. The fingerprinting methods calculate the hash value of the tested documents and compare it with the the hash value of trained template. The detection phase will be accurate and fast if the file is the original file. But once the file makes any change, its fingerprinting will be completely different from the original one, making it hard to detect. Some work is devoted to improve the method for robust fingerprinting. Blocking hash is used instead of overall hash to tolerate the text transformation [13].

The statistical methods are mainly based on the statistical features of the text (e.g, terms and their frequencies). The word n-gram based classification method is another type of content analysis based DLD method, which calculates the offset of the grams between the tested file and the template file, sorted by their frequencies [7]. The smaller the offset, the more sensitive the tested file is. But it is incomplete to represent a document only by its high frequency terms. To better choose the key terms to represent a document, term weighting is used to indicate the semantic importance of the word. Term frequency-inverse document frequency (TF-IDF) is a best known method used to detect the data semantic to prevent data leakage [14].

Machine learning classification methods are well explored in areas of information retrieval, text categorization and spam filtering, such as Naive Bayes [8] and support vector machines (SVM). These methods also have limitations in DLD. An automatic text classification method based on SVM is proposed in [9]. It maps the text into vector space, and trains the classifier with the features of terms and their frequencies. The documents are then classified as either sensitive or non-sensitive. However, different from text classification tasks, the number of sensitive files in training set is usually very small while the non-sensitive is large, the machine learning methods mentioned above will learn the most significant statistical features, ignoring the sensitive context weight of terms, which will lead to a high false alarm rate.

All in all, the content analysis methods such as rule-based methods and fingerprinting methods are hard to deal with data transformation, while the statistical methods only learn the most statistical characteristics, ignoring the sensitive context weight. However, with the development of Internet and the emerging of cloud service providers (CSPs) [15], a large number of data in various forms are produced and need to be monitored. The above methods are hard to cope with these transformed data.

B. CONTEXT ANALYSIS

The recently studied context analysis methods attempt to take the sensitive context of terms into consideration. The adaptable n-gram [16] is proposed to classify documents into different detailed sensitive topics according to a simple overall statistical context. The improved n-gram method [11] is also proposed by using subsequence-preserving sampling to preserve some of the context information. Although these methods compensate some limitations of traditional n-gram methods, they still consider little or very simple context information in data leak detection. Some studies have already tried to use graphs to represent the text [17]–[19] in summarization, information retrieval and DLD tasks instead of using word vectors, which has yielded good performance. An improved approach called CoBAn [12] reduces the false positives by considering the context of key terms in graphs for classification. CoBAn has achieved good performance. It assigns one from a set of clusters to match a document. But the context it defined is too rigid to satisfy in the scene of processing transformed data. In addition, the computational complexity of existing context analysis methods can still be further improved to accommodate big data scenarios.

Recently, some new challenges are faced by DLD with transformation of data. Research work has been done to handle the detection of transformed data. The word n-gram based classification method proposed in [7] tested the cases of document modification by subtracting and adding words to the original text body and most cases can be classified correctly. The AlignDLD method proposed in [11] can detect modified leaks caused by WordPress, which substitutes every space with a "+". The CoBAn [12] also can detect the documents out with modified sensitive sections. However, only simple transformation scenarios are considered in most of the papers. The assumed data transformation is either embedding some sensitive content in the original text or replacing some of the original text. How to tolerate the data transformation to a greater degree is still a huge challenge in DLD.

In this paper, AGW is proposed to handle extensively transformed data. We use the adaptive context weight mechanism to better preserve the information of key terms and their context, which can tolerate a large degree of data transformation. We also propose a low-complexity algorithm with a weight reward and penalty mechanism, to deal with the large amounts of data in big data scenario. Thus, AGW can improve the detection accuracy towards transformed data and has a low computational complexity.

III. MODEL

In this section, the problem formulation of data leak detection in this paper is given at first. Then we present the overview and details of our solution and its analysis. The proposed AGW consists of two phases: adaptive weighted graphs learning and score walk matching detection. The adaptive weighted graphs are learned from the training set, retaining the key terms and their context characteristics to represent a document comprehensively. The score walk detection analyzes the tested documents and matches them to the trained template and to determine whether the tested documents are sensitive. The details are described in the following subsections.

A. PROBLEM FORMULATION

Data leakage happens when data is distributed from the inside of an organization to unauthorized outsiders, intentionally or accidentally, which may cause serious damage to the organization. Considering such a common scenario: an employee of an organization tries to leak sensitive files from local network to unauthorized outsiders. To prevent data leakage, a detection agent is needed to inspect the outgoing traffic content. The detection agent is usually deployed at the outlet of the local network. The overall application scenario is shown in Fig. 1.

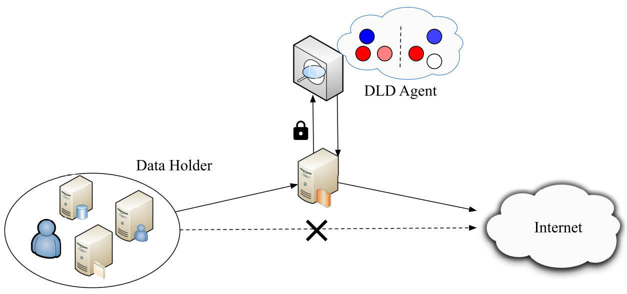


FIGURE 1. The overview application scenario of AGW.

Given a dataset D as the training set with documents $d_1, d_2, \ldots, d_m$. The documents contain two categories: sensitive D_S and non-sensitive D_N, with $\mathbb{R} = \{D_S, D_N\}$ to represent the classified dataset. $D' = \{d_1, d_2, \ldots, d_n\}$ is an unclassified dataset, denoting the outgoing data to be tested, which means whether the documents in D' are sensitive or not is unknown. The goal of this paper is to establish a mapping $\mathcal{M} : D \to \mathbb{R}$ on D. Then the mapping $\mathcal{M}$ can be used to build $D' \to \mathbb{R}'$, where $\mathbb{R}' = \{D'_S, D'_N\}$. Thus, sensitive files in D' can be detected to determine if there is a data leak.

The problem becomes how to classify the test set D' into $\mathbb{R}'$: $D' \to \mathbb{R}'$ based on training set D. The key point to solve this problem is to build a appropriate mapping model $\mathcal{M}$. For any $d_i \in D'$, there are three possible cases: 1) $d_i \in D$, means there exist one $d_j \in D$, that $d_j = d_i$; 2) $d_i \notin D$, but there exist a $d_j \in D$ which is partly similar to d_i; 3) $d_i \notin D$, nor is there any $d_j \in D$ that is similar to d_i. The above cases illustrate two challenges need to be tackled in this scenario:

- How to handle the $d_i \in D'_S$ in case 2, that is, how to detect the transformed sensitive data?
- How to deal with the $d_i \in D'_S$ in case 3, which means how to detect the completely transformed unknown data?

What's more, the scale of data is usually large in Big Data scenarios, how to perform the detection more efficiently also deserves much consideration.

B. OVERVIEW OF AGW

We address the above challenges in AGW. AGW mainly contains two phases: adaptive weighted graphs learning and score walk matching detection. The adaptive weighted graphs learning phase builds the template graphs from training set. It retains the key information of the training set as much as possible. The score walk matching detection calculates the sensitivity scores of tested documents, and determine if there is a data leak based on the sensitivity scores. The learning phase can be performed locally at the data holder's side, while the detection phase can be performed by one or several third party agents. What's more, the DLD agent may be semi-honest for some reasons (e.g, attacked or controlled by malicious users), which may try to learn the sensitive data from data holder. To figure this problem out, we apply privacy-preserving graph masking before the sensitive data is sent to the agent. The graph masking is performed respectively after the template graphs are constructed in the learning phase and the tested graphs are constructed in the score walk matching phase. The original content of the graph nodes (sensitive terms) can be concealed by using graph masking. The adaptive weighted graphs learning, the privacy-preserving graph masking and the score walk matching constitute the key modules of AGW. The overview of AGW is shown in Fig. 2.

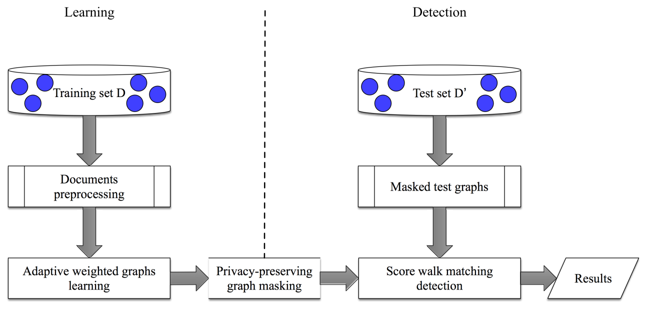

Learning
Detection
Training set D
Test set D'
Documents preprocessing
Masked test graphs
Adaptive weighted graphs learning
Privacy-preserving graph masking
Score walk matching detection
Results

FIGURE 2. The overview of AGW.

C. ADAPTIVE WEIGHTED GRAPHS LEARNING

The word vector space model represents text by vectors, which is widely used in many text-related tasks, such as information retrieval and sentiment analysis. The graphs are another method which also has been used for text representation in many text analysis tasks. Different from the tasks of natural language processing (NLP), the sensitivities of terms are more concerned than their meanings in data leak detection. Graphs can better keep the structure information of a document besides its content. We use adaptive weighted graphs to retain the sensitive semantic of documents.

Fig. 3 shows the learning phase of adaptive weighted graphs. We define the adaptive weighted graph based on traditional graphs in Definition 1.

Definition 1 (Adaptive Weighted Graphs): Let V be the set of nodes (key terms), and each node has a value of text term and a weight w_n. The set of node weights is W_N.

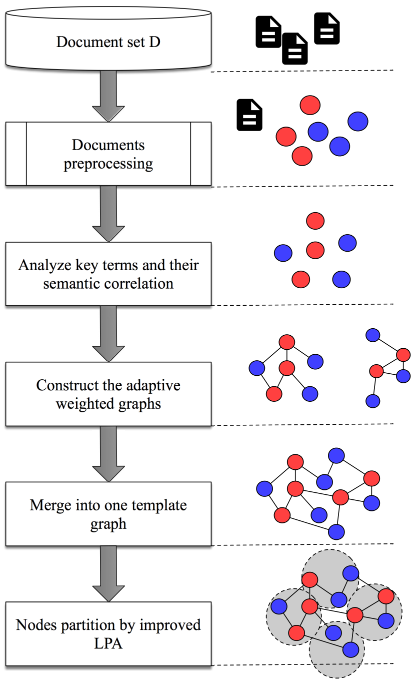

Document set D
Documents preprocessing
Analyze key terms and their semantic correlation
Construct the adaptive weighted graphs
Merge into one template graph
Nodes partition by improved LPA

FIGURE 3. The learning of adaptive weighted graphs.

An edge e connecting two nodes denotes that they have a certain degree of correlation, and each edge has a weight w_e to quantify this correlation degree. Let W_E be the set of edge weights. Weight matrix A consists of elements $w \in W_N \cup W_E$, which contains the entire structure and weight information of the graph. An adaptive weighted graph is denoted as $G = \{V, A\}$.

An adaptive weighted graph is a summary of a document which keeps the key terms and their contextual information of the document. In the learning phase, how to retain as much key information as possible and simplify the graph to minimize the amount of subsequent calculation is the main concern. The adaptive weighted graph provides a balance between the two points mentioned above. Moreover, what DLD concerns with is not the word semantic, but the sensitivity of the content. We call it sensitivity semantic to distinguish from the words semantic used in NLP. The sensitivity semantic defined in AGW focuses more on the appearance of key terms and their contextual correlations.

The complete detailed steps of the weighted graphs learning phase are described below.

1) DOCUMENTS PREPROCESSING

The adaptive weighted graphs learning starts with documents preprocessing. There are two categories of documents: sensitive and non-sensitive. Documents are first "cleaned" by removing the stop words and any meaningless words(e.g., 'the' and 'ing' suffix). Then the documents are segmented into terms of various lengths, where each term may contain one or more words.

2) ANALYZING KEY TERMS AND THEIR SEMANTIC CORRELATIONS

In the second step, the nodes and edges of weighted graphs are established. Since the frequencies of terms are important characteristics of documents, the TF-IDF has achieved good performance in extracting the representative features of files. We adjust the TF-IDF method to better consider sensitivity semantic. Given two sets of segmented documents: sensitive D_S and non-sensitive D_N. Let $d \in D_S$ denotes a segmented file in sensitive files, $n_{t,d}$ denotes the number of times that term t occurs in document d. The term frequency is

$$tf(t,d) = \frac{n_{t,d}}{\sum\limits_{t' \in d} n_{t',d}}, \tag{1}$$

where the $\sum\limits_{t' \in d} n_{t',d}$ denotes the sum number of all terms in d. The inverse document frequency IDF is

$$idf(t, D_N) = log \frac{|D_S|}{1 + |\{d' \in D_N : t \in d'\}|}, \tag{2}$$

where $|D_S|$ is the total number of sensitive documents, $|\{d' \in D_N : t \in d'\}|$ is the number of non-sensitive documents where the term t appears. Thus the weight w (adjusted TF-IDF) of term t is defined as Eq. (3), which reflects the sensitivity of term t in the document.

$$w_n = tf(t,d) \cdot idf(t, D_N) \tag{3}$$

The key terms of one document are then analyzed according to the weight w of each term. Terms with top N weights will be selected out as the nodes by a selector f,

$$V = f(D, N) \tag{4}$$

where V is the set of nodes defined in definition 1. An edge e between two nodes will be set only when the distance n_e between them is less than a threshold K, where n_e is the number of terms between the two nodes in the segmented document d.

3) CONSTRUCTING THE ADAPTIVE WEIGHTED GRAPHS

In this step, the adaptive weighted graphs are constructed. With nodes and edges set in last step, the node weights and edge weights are then calculated. Eq. (5) provides the edge weight of e.

$$w_e = \frac{K - n_e}{K} \tag{5}$$

w_e quantifies the degree of correlation between nodes. The weight matrix A is defined in Eq. (6). It contains the weight information of nodes and their edges, which describes the context structure of the weighted graph. Each row of A represents an node in ranked sequence, and so is the column with the same nodes sequence. The w_{n_i} on the diagonal denotes the node weight w_n of node i. The $w_{e_{ij}}$ denotes the edge weight of the edge between node i and node j, and $w_{e_{ij}} = 0$ if there is no edge between node i and node j.

$$A = \begin{bmatrix} w_{n_1} & w_{e_{12}} & \cdots & w_{e_{1n}} \\ w_{e_{21}} & w_{n_2} & \cdots & w_{e_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{e_{m1}} & w_{e_{m2}} & \cdots & w_{n_n} \end{bmatrix} \tag{6}$$

At this point, the node set V together with the weight matrix A, form a weighted graph $G_d = \{V_d, A_d\}$.

4) MERGING INTO ONE TEMPLATE GRAPH

In the final step, the weighted graphs are merged into one template graph $G(V, A)$ through Eq. (7).

$$\begin{aligned} G(V, A) &= G(V_{d_1}, A_{d_1}) \cup \cdots \cup G(V_{d_i}, A_{d_i}) \\ &= G(V_{d_1} \cup \cdots \cup V_{d_i}, A_{d_1} \cup \cdots \cup A_{d_i}) \end{aligned} \tag{7}$$

If two or more nodes belonging to different graphs have the same text value, the merged weight will be an mean

$$w_n = \frac{1}{n} \sum_{i=1}^{n} w_{n_i}. \tag{8}$$

So is the same if two or more edges connect two same nodes, the mean merged edge weight is

$$w_e = \frac{1}{n} \sum_{j=1}^{n} w_{e_j}. \tag{9}$$

5) NODES PARTITION BY IMPROVED LPA

For the correlations of template graph are derived directly from the sub-graphs, they may be incomplete and one-sided, which can not reflect the statistical correlations between nodes. We solve this problem by using improved label propagation algorithm to conclude and generalize the overall correlations based on the original edges. We re-partition the template graph by dividing the nodes into different categories. The label of a node indicates which category it belongs to. Label propagation is an semi-supervised learning method initially proposed for community detection in networks [20]. It is a heuristic method which can handle a large number of unlabeled data with a small amount of annotation data. With a concise core idea and good scalability, label propagation is one of the simplest and fastest methods for dealing with large graph. We improved the original label algorithm to make it more suitable for processing our template graph. The main idea is that a node's label is always the same with one of its neighbor nodes' label which owns maximum label weight. The weight of label l for node i is denoted by $w_{i,l}$, which sums all the transition probability $P(j \to i)$ of nodes adjacent to i. The $w_{i,l}$ is shown in Eq. (10).

$$w_{i,l} = \sum_{l,j} P(j \to i) = \sum_{l,j} \frac{w_{n_j} + w_{e_{ij}}}{\sum_j (w_{n_j} + w_{e_{ij}})} \tag{10}$$

The improved label propagation is shown in Algorithm 1.

Algorithm 1 The Improved Label Propagation

```
Input: template graph $G(V, A)$, stop criteria $\eta$, maximum
    number of iterations $C$
Output: labeled template graph $G(V, A, L)$
1: Initialize: $L = \emptyset$, iterations $t = 0$, randomly assign a
   label to each node
2: while do not meet $\eta$ and iterations $t < C$ do
3:   while node $i$ in $G$ do
4:     calculate the label of node $i$: select the label with
       maximum $w_{i,l}$ from its neighbors
5:   end while
6: end while
7: return $G(V, A, L)$
```

The complete algorithm of the graph learning phase is shown in Algorithm 2. The sensitive template graph G_S is learned from sensitive training set, $D_S \rightarrow G_S$, as well as the non-sensitive template $D_N \rightarrow G_N$.

Algorithm 2 The Graph Learning

```
Input: set of documents $D$, a selector $f(d, N)$ that selects the
    top $N$ representative terms from document $d$
Output: graph $G(V, A, L)$ - nodes set $V$, weight matrix $A$
    and category labels $L$
1: Initialize: $V = \emptyset, A = O, L = \emptyset$
2: while $d$ in $D$ do
3:   Process $d$, remove the meaningless words and segment
     $d, d \rightarrow d^{seg}$
4:   Calculate the adjusted TF-IDF of each term, select the
     most representative N terms as nodes, $V = f(d^{seg}, N)$
5:   Get weight set of $V$, compute weight matrix $A$
6:   Get $G_d = (V_d, A_d)$
7: end while
8: Merge $G_d \rightarrow G$: compute the merged weights $W$ and
   merged weight matrix $A$
9: Partition the nodes into various categories by
   improved label propagation in Algorithm 1,
   $G(V, A) \rightarrow G(V, A, L)$
10: return $G(V, A, L)$
```

D. PRIVACY-PRESERVING GRAPH MASKING

As mentioned before, the DLD agent is considered as semi-honest in our model. It is a potential threat to the data holder if the agent can come into contact with the original sensitive data. Thus, how to ensure that the agent can perform the detection efficiently without learning any sensitive data is a significant problem.

In AGW, we apply the proposed privacy-preserving graph masking method on the original weighted graphs to figure this problem out. The goal of graph masking is to keep the value of the nodes (the content of sensitive terms) from being learned by the agent. The masking process is carried out before the weighted graphs are sent to the agent. The weighted template graph is constructed locally at the data holder's side, consisting of nodes set and weight matrix. We apply the irreversible encryption (hash function) to the original value of each node to encrypt the content of sensitive terms, while the weights and edges correlations are preserved. The masked template graphs, containing nothing about the sensitive content, will be provided to the detection agent from the data holder. The agent performs the detection based on the masked template graphs as profiles. Since our detection mainly focus on value matching, the encryption will not affect the performance of detection. The overall masking process is shown in Fig. 4.

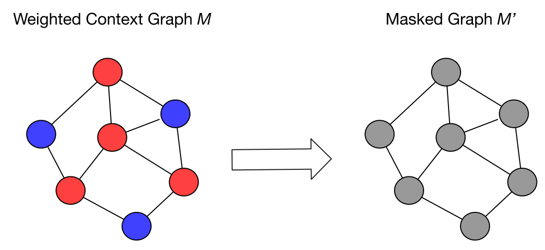


FIGURE 4. The process of graph masking.

The privacy-preserving graph masking method can keep the sensitive content unknown to the detection agent, while the structures and correlations are preserved for further detection. The mechanism of AGW ensures the matching process unaffected by the graph masking. Therefore, AGW can perform the detection on the masked graphs, without affecting the original detection effect.

E. SCORE WALK ALGORITHM

A score walk algorithm is proposed in this section to detect the sensitive data. In the detection phase, there are mainly two challenges: 1) how to detect the large amounts of transformed data? 2) how to detect the large amounts of fresh data? We transfer these two problems to a graph matching problem in AGW. The two graphs may constructed from two similar documents, but one of which may be transformed or fresh to the other. How to match such two graphs to further detect sensitive data is the main problem we need to tackle.

Fig. 5 presents the flow of detection phase. Since the goal of detection is to calculate the sensitivity of each document, the graphs should be tested separately instead of being merged. The documents to be tested are first processed according to graphs learning Algorithm 2 except step 8, $D' \rightarrow G'$, where $G' = \{G'_{d_1}, \ldots, G'_{d_n}\}$ is the set of weighted graphs. Then the improved label propagation proposed in Algorithm 1 is applied on each graph G'_d respectively. Nodes belong to both G'_d and template G are first initialized by their category label in G, while the others are initialized randomly. The rest of improved label propagation is performed as in Algorithm 1. At this stage, all the tested documents have been converted into weighted graphs.

The score walk algorithm is proposed to solve the problem of graph matching between tested graphs G' and template graph G. For each graph $G'_d \in G'$ to be tested, its nodes

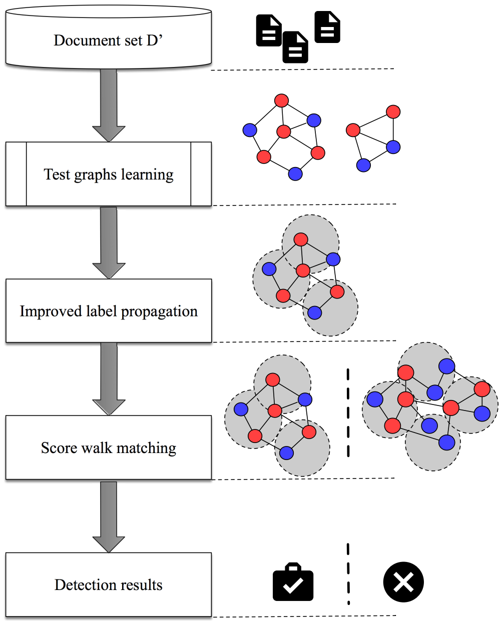


FIGURE 5. **The detection phase.**

and edges are walked through by breadth-first traversal to calculate a matching score. A reward and punishment mechanism is used to calculate the score $f_w()$ during the process of walking. For a node $n_i \in G'_d$, if $n_i \in G$, a node bonus $f_w(n_i, +)$ as shown in Eq. (11a) will be added. Then all nodes n_j connecting to n_i through edge e_{ij} will be traversed. If an edge $e_{ij} \in G$, an edge bonus $f_w(e_{ij}, +)$ as shown in Eq. (11b) will be added. If $e_{ij} \notin G$ but in G there exist one node also connecting to n_i with the same label as $n_j.label$ in G', a label bonus $f_w(e_{ij}, lb, +)$ as shown in Eq. (11c) will be added. Otherwise a label penalty $f_w(e_{ij}, lb, -)$ as shown in Eq. (11d) will be subtracted. If $n_i \notin G$, a node penalty $f_w(n_i, -)$ as shown in Eq. (11e) and its edges penalty $f_w(e_{ij}, -)$ as shown in Eq. (11f) will be together subtracted, to punish the mismatched node and its correlations. The detailed rewards and penalties mentioned above are listed in Eq. (11).

$$f_w(n_i, +) = w_{n_i} + \alpha \cdot (\frac{w_{n_i}}{w^T_{n_i}}) \tag{11a}$$

$$f_w(e_{ij}, +) = w_{e_{ij}} + \beta \cdot (\frac{w_{e_{ij}}}{w^T_{e_{ij}}}) \tag{11b}$$

$$f_w(e_{ij}, lb, +) = w_{e_{ij}} + \beta \cdot w_{e_{ij}} \cdot \gamma \tag{11c}$$

$$f_w(e_{ij}, lb, -) = \beta \cdot w_{e_{ij}} \cdot \gamma \tag{11d}$$

$$f_w(n_i, -) = \alpha \cdot w_{n_i} \tag{11e}$$

$$f_w(e_{ij}, -) = \beta \cdot w_{e_{ij}} \tag{11f}$$

The whole graph G'_d is traversed by the score walk, during which the scores of nodes and edges are calculated in turn and added up. The total sensitivity score of G'_d is shown in Equation. 12, where $(-1)^{k_1} f_w(n_i, k_1)$ denotes the node reward ($k_1 = 2$) or penalty ($k_1 = 1$), $(-1)^{k_2} f_w(e_{ij}, k_2)$ denotes the reward($k_2 = 2$) or penalty ($k_2 = 1$) of edge connecting n_i and n_j.

$$f_w(G'_d) = \sum_{\substack{n_i \in G'_d, \\ e_{ij} \in G'_d}} (-1)^{k_1} f_w(n_i, k_1) + (-1)^{k_2} f_w(e_{ij}, k_2) \tag{12}$$

Algorithm. 3 provides the detail of score walk algorithm. The sensitivity score of a weighted graph is calculated by one time score walk on the graph, which can perform the detection quickly and efficiently. The final sensitivity is determined by comparing the two score toward sensitive template G_S and non-sensitive template G_N. What's more, through the mechanism of reward and punishment, it can perform a general match between the tested graphs and template graph, which can tolerate the transformed or fresh nodes and edges in tested graphs. Thus, score walk algorithm can deal with the large amounts of transformed data and fresh data.

Algorithm 3 The Score Walk

```
Input: weighted graph $G'_d$, template graph $G$
Output: The sensitivity of graph $G'_d$ - $f_w(G'_d)$
1:  Initialize: $f_w(G'_d) = 0$
2:  Breadth-first traversal $G'_d$
3:  while node $n_i$ in $G'_d$ do
4:    if $n_i \in G$ then
5:      $f_w(G'_d) = f_w(G'_d) + f_w(n_i, +)$
6:      while $e_{ij}$ connects $e_i$ and $e_j$ do
7:        if $e_{ij}$ in $G$ then
8:          $f_w(G'_d) = f_w(G'_d) + f_w(e_{ij}, +)$
9:        else
10:         if $n_j.labelinG$ then
11:           $f_w(G'_d) = f_w(G'_d) + f_w(e_{ij}, lb, +)$
12:         else
13:           $f_w(G'_d) = f_w(G'_d) - f_w(e_{ij}, lb, -)$
14:         end if
15:       end if
16:     end while
17:   else
18:     $f_w(G'_d) = f_w(G'_d) - f_w(n_i, -)$
19:     while $e_{ij}$ connects $e_i$ and $e_j$ do
20:       $f_w(G'_d) = f_w(G'_d) - f_w(e_{ij}, -)$
21:     end while
22:   end if
23: end while
24: return $f_w(G'_d)$
```

F. COMPLEXITY ANALYSIS

The AGM consists of two phases: template graph learning and score walk detection. The template graph learning is executed at the data holder's side asynchronously and off-line, while the score walk detection is carried out at the detection agent in real-time. So the complexity of the detection phase is much more important than the learning phase, which is directly related to the efficiency of the real-time detection. We mainly analyze the complexity of the score walk detection. For a document d to be tested, the score walk detection mainly has two steps: (a) learning the weighted graph G'_d, (b) score walk detection of G'_d.

To analyze the complexity of detection phase, we assume that the template graph G has already been constructed. The weighted graphs are stored in the HashMap, whose cost of looking up a value is $O(1)$. Assuming there are C terms in document d, the top N terms will be selected out as nodes, the window size of establishing an edge is K, and the number of edges is M. The complexity of step (a) and (b) are as follows:

- learning of G'_d: each word of document d is read once to record its count and index position. The cost of this process is $O(C)$. Then the adjusted TF-IDF value of each term is calculated at the cost of $O(C)$. The top N terms are selected out as nodes. For each node, its adjacent nodes are searched in the window of K, the complexity is $O(N \cdot K)$. In label propagation process, the initialization of each node label requires $O(N)$ time, and each iteration of label propagation algorithm takes linear time in the number of edges $O(M)$ [21]. The overall complexity is $O(2C+N\cdot K+N+M)$. For each node in G'_d, the number of its edges is less than K, as its edges are set in the window of K. Thus, $M < K \cdot N$, the complexity can be reduced to $O(K \cdot N)$.
- detection of G'_d: the sensitivity score of graph G'_d is calculated through one time score walk on G'_d. This process takes liner time in the sum of the number of nodes and edges, $O(N + M)$. For $M < K \cdot N$, the complexity can be simplified to $O(K \cdot N)$.

To sum up, the complexity is $O(2K \cdot N) = O(K \cdot N)$. The steps described above will be applied on each document d, where $d \in D'$. These repeated calculation takes a linear time to the complexity of each document. Thus, the overall complexity is $O(K \cdot N)$. As we can see, the overall complexity is liner to the number of nodes and the size of edge window, while the two parameters are both constant values in our implementation. Therefore, it can be concluded that AGW is capable of being implemented in real time.

IV. EVALUATION

In this section, the performance of AGW is evaluated by a series of experiments.

A. IMPLEMENTATION AND EXPERIMENT SETUP

For there are few published sensitive datasets available due to the confidentiality of sensitive data, the Reuters dataset [22] is adopted as the raw dataset to perform our experiments. The dataset is a series of articles appeared on Reuters newswire, which consists of 11592 training documents and 4140 testing documents in 116 categories, each category in one folder. It is also a multi-label dataset which means each document may belong to many classes. It must be figure out that data leak detection is different from the task of document classification. What DLD do is to assess the sensitivity of documents according to sensitivity semantic. Thus, it concerns the exact category of the documents, while the multi-label cases will be ignored by DLD. Since many categories in the basic dataset are too small (e.g., one or two documents only) to perform further evaluation, we select the categories out whose number of documents is more than 100, both in the training set and test set. 7 qualified categories (the number of documents ranges from hundreds to thousands) are screened out from the 116 categories as the basic dataset. The basic dataset consists of 6611 training documents and 2579 testing documents. To better simulate the real-world data leak scenarios, we specify the sensitive data by assuming one category as sensitive and the others as non-sensitive. For more detailed further evaluation, we combine one category as sensitive and another as non-sensitive randomly from the basic dataset each time. Finally, 42 sets of data are generated for the following statistical experiment, each with a training set and a test set containing thousands of documents.

TABLE 1. The confusion matrix.

	Actual sensitive	Actual non-sensitive
Predicted sensitive	TP	FP
Predicted non-negative	FN	TN

Among the detection phase, we use differential score $S_d = S_{d,G_S} - S_{d,G_N}$ instead of a fixed score threshold to determine the final sensitivity of document d, where S_{d,G_S} denotes the calculated score of d towards sensitive template G_S, while S_{d,G_N} towards G_N. A tested document d is determined to be sensitive only when $S_d > 0$. We use the relative differential score other than a fixed threshold to avoid the impact of a threshold on the final detection results.

The effect of detection is measured by accuracy, recall and receiver operating characteristic (ROC) curve. The confusion matrix in Table. 1 presents the detailed analysis parameters of detection results. The accuracy (Acc) measures the rate of correctly detected cases, calculated by $Acc = \frac{TP+TN}{TP}$. The recall measures the rate of sensitive documents that are also detected as sensitive, calculated by $Recall = \frac{TP}{TP+FN}$, which reflects the ability to detect sensitive data leaks. The ROC curve is plotted with TP against FP at various threshold, illustrating the diagnostic ability of our proposed model.

The experiments involve 42 sets of data, which range in size from a few hundred to several thousand. Scenarios of detecting unmodified data and detecting transformed data are tested, with different sizes of dataset. The documents of same category in test set are quite different from the training set, so the test set is used as the modified or fresh data to be detected. The following experiments in this section aim to answer the following questions.

- Can the proposed method deal with large amounts of data efficiently?
- Can the proposed method tolerate the transformed or fresh data in detection?

TABLE 2. The accuracy and recall of 42 tested groups.

Number of groups	Accuracy	Recall
40	100%	100%
1	99.91%	100%
1	91.64%	98.51%

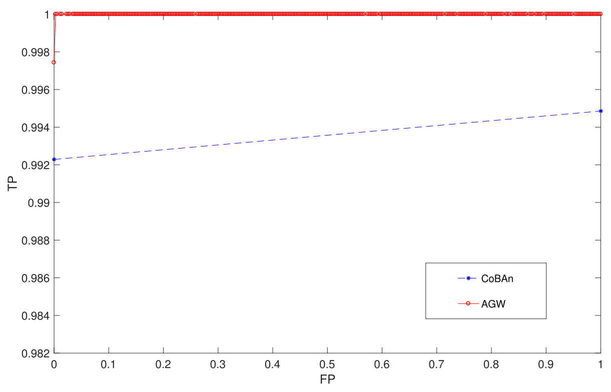


FIGURE 6. The ROC curves of CoBAn and AGW.

- How does our method compare to the state-of-the-art method on the same dataset in terms of accuracy, recall, ROC?

The performance of AGW in two scenarios is evaluated in our experiments. Firstly, experiments for detecting unmodified data have been done as the basic scenario. Then, experiments for detecting modified data have been done to find out how AGW performs when faced with transformed data. This simulates the real-world transformed data leakage scenario.

B. DETECTING UNMODIFIED DATA

In the scenario of detecting unmodified data leaks, the training set of the basic dataset is used to learn the template weighted graph as described in Algorithm 2, which is also used as the test dataset. The statistical results of 42 test sets are shown in Table 2. Among the 42 test groups, the accuracy and recall of 40 groups are both 100%, while the other two groups are 99.91%, 100%, and 91.64%,98.51%. The results show the strong ability of AGW in detecting unmodified data leaks.

Fig. 6 presents the ROC curves of AGW and CoBAn [12]. Both of these two methods have achieved high performance in ROC curve, while AGW is a little higher. The high performance is as expected for the DLD task here is similar to a basic classification problem in the unmodified scenario. It indicates that to detect the unmodified data leakage is much easier because the data to be tested changes little from the template data.

The training time, testing time and total time of AGW are shown in Fig. 7. The time increases linearly as the size of dataset increases, which is consistent with our previous complexity analysis in Section III. It can also be seen from Fig. 7 that the running time of test phase is much less than the training phase. As mentioned before, the training phase is performed locally and off-line, while the test phase has a high demand for time efficiency. It can be concluded from the results that the proposed method can deal with large amounts of data efficiently.

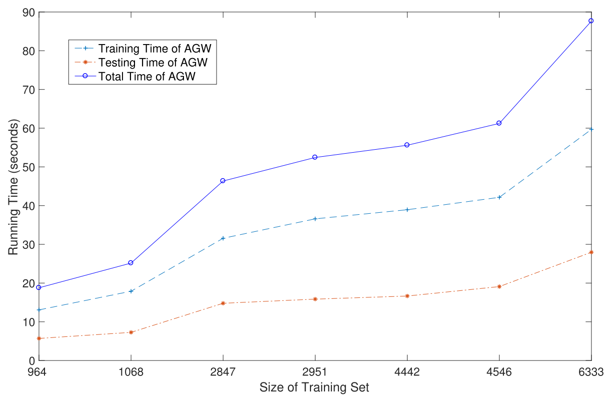


FIGURE 7. The running time of AGW in different sizes of test set.

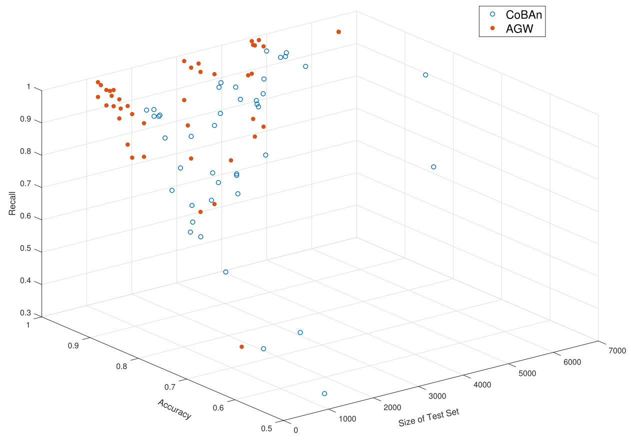


FIGURE 8. The accuracy and recall of CoBAn and AGW in 42 test sets.

C. DETECTING TRANSFORMED LEAKS

In order to test the capability of our method in tolerating data transformation, the transformed data should be used as the data to be detected. Instead of simply adding some content or partly modifying the original documents, the documents in the test set are completely different and new compared with those in our training set. For sure they are in the same category. The new documents in our test set are chosen as the transformed sensitive data to verify the ability of AGW to detect completely transformed data. A series of datasets with various size have been tested. The overall statistical results of all 42 test sets are presented in Fig. 8. As we can see, among the 42 groups, AGW performs better in most cases.

More specifically, Fig. 9 shows the average detection accuracy of AGW and CoBAn, denoting the rate of correctly classified (sensitive or not) cases. The recalls of AGW and CoBAn are provided in Fig. 10. It is a proportion of documents that correctly detected as sensitive in total sensitive documents, denoting the capability of the method to detect

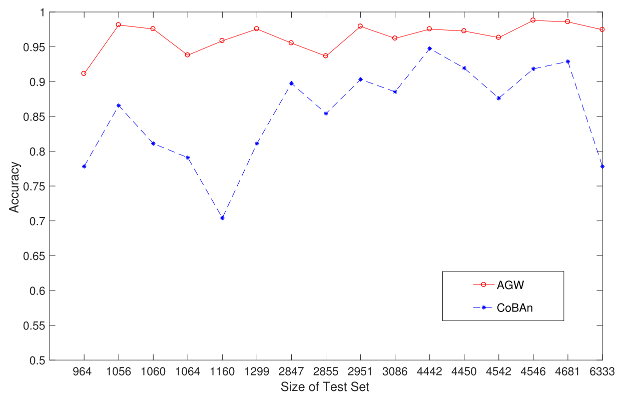


FIGURE 9. The accuracy of CoBAn and AGW in different sizes of test set.

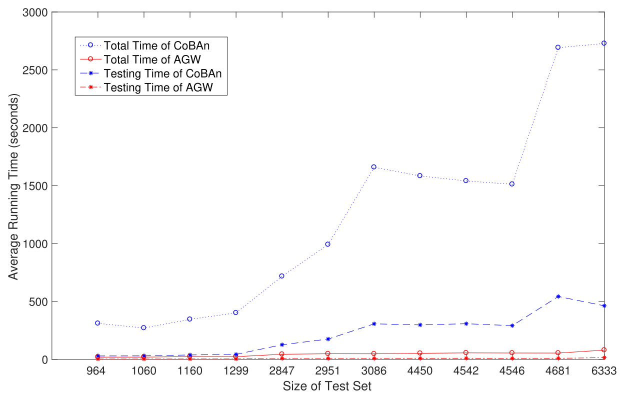


FIGURE 11. The running time of CoBAn and AGW.

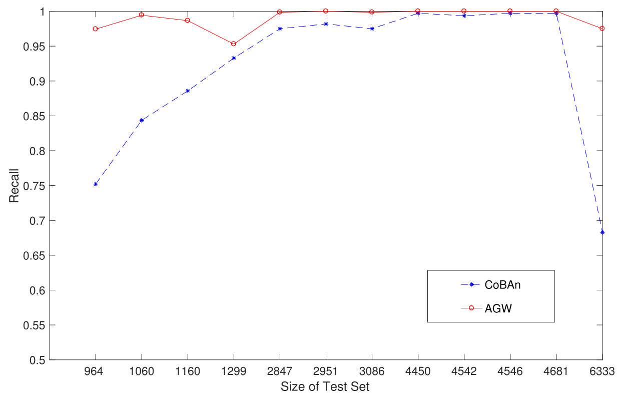


FIGURE 10. The recall of CoBAn and AGW in different sizes of test set.

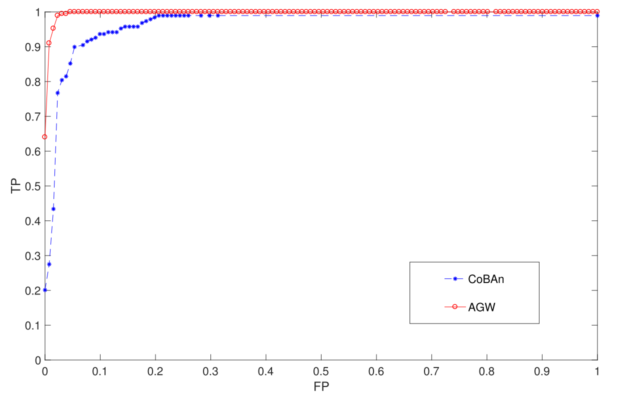


FIGURE 12. The ROC curves of CoBAn and AGW.

the leaks of sensitive data. As denoted in Fig. 9, Fig. 10, AGW achieves a higher detection accuracy and recall on various test sets with different sizes. The accuracy and recall of AGW change little with the increase of the size of test set, while the accuracy and recall of CoBAn fluctuate widely in some cases. This difference may be due to that the documents of same category in the test set and training set are quite different. AGW catches more correlation information of key terms in the document by using weighted edges, while CoBAn uses only context terms to represent the correlation, which is hard to satisfy. The weighted context graph of AGW is more suitable for dealing with transformed data.

The total running time and detection time (in seconds) of CoBAn and AGW are provided in Fig. 11. As is shown, AGW outperforms CoBAn in terms of total running time and test time. What's more, the running time of AGW increases slowly as the size of test set increases, whose curve is close to flat. This performance is consistent with previous analysis in Section III that the complexity of AGW is $O(K \cdot N)$, which is liner to the number of key terms in a document and the size of edge window, and has little to do with the size of document. While CoBAn performs the detection in a complexity of $O(T \cdot C)$ [12], which is liner to the number of terms (T) in a document and the number of clusters (C).

The ROC curves of CoBAn and AGW are also presented in Fig. 12. The TP axis denotes the rate of real sensitive cases still detected as sensitive in test results, while FP denotes the "false alarms" in sensitive results.

The experimental results presented above provide an encouraging answer that AGW can detect various amounts of transformed data leaks efficiently, ranging from kilobytes to megabytes.

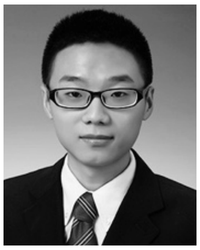
V. CONCLUSION

In this paper, we present a new method for data leak detection named AGW. Our method consists of two phases: adaptive weighted graphs learning and score walk matching. The proposed method mainly has the following three contributions: First, it can tolerate the long transformed data by using weighted context graphs. Second, It increases the scalability to deal with fresh data by applying improved label propagation algorithm. Third, it can detect large amounts of data with a weight reward and penalty mechanism named score walk matching. The evaluation shows the effectiveness of AGW in terms of accuracy, recall and running time. It can be concluded that our method can perform the fast detection for transformed or fresh data leaks in real time.

REFERENCES

[1] S. Alneyadi, E. Sithirasenan, and V. Muthukkumarasamy, "A survey on data leakage prevention systems," *J. Netw. Comput. Appl.*, vol. 62, pp. 137–152, Feb. 2016.
[2] Wikileaks. (2010). *Secret US Embassy Cables*. [Online]. Available: https://wikileaks.org/plusd/
[3] B. News. (2013). *Edward Snowden: Leaks That Exposed US spy Programme*. [Online]. Available: http://www.bbc.com/news/world-us-canada-23123964
[4] Wikipedia. (2016). *Yahoo! Data Breaches*. [Online]. Available: https://en.wikipedia.org/wiki/Yahoo!_data_breaches
[5] J. Huang, X. Li, C.-C. Xing, W. Wang, K. Hua, and S. Guo, "DTD: A novel double-track approach to clone detection for RFID-enabled supply chains," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 1, pp. 134–140, Jan./Mar. 2017.
[6] K. Lab. (2016). *It Security Risks Report 2016*. [Online]. Available: https://www.kaspersky.com/blog/security_risks_report_perception
[7] S. Alneyadi, E. Sithirasenan, and V. Muthukkumarasamy, "Word N-gram based classification for data leakage prevention," in *Proc. 12th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Jul. 2013, pp. 578–585.
[8] S. Xu, "Bayesian Naïve Bayes classifiers to text classification," *J. Inf. Sci.*, vol. 44, no. 1, pp. 48–59, 2016.
[9] M. Hart, P. Manadhata, and R. Johnson, "Text classification for data loss prevention," in *Proc. Int. Symp. Privacy Enhancing Technol. Symp.* Berlin, Germany: Springer, 2011, pp. 18–37.
[10] J. Huang, S. Huang, C.-C. Xing, and Y. Qian, "Game-theoretic power control mechanisms for device-to-device communications underlaying cellular system," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 4890–4900, Jun. 2018.
[11] X. Shu, J. Zhang, D. D. Yao, and W.-C. Feng, "Fast detection of transformed data leaks," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 3, pp. 528–542, Mar. 2016.
[12] G. Katz, Y. Elovici, and B. Shapira, "CoBAn: A context based model for data leakage prevention," *Inf. Sci.*, vol. 262, pp. 137–158, Mar. 2014.
[13] X. Shu, D. Yao, and E. Bertino, "Privacy-preserving detection of sensitive data exposure," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 5, pp. 1092–1103, May 2015.
[14] S. Alneyadi, E. Sithirasenan, and V. Muthukkumarasamy, "Detecting data semantic: A data leakage prevention approach," in *Proc. Trustcom/BigDataSE/ISPA*, vol. 1, Aug. 2015, pp. 910–917.
[15] J. Huang, J. Zou, and C.-C. Xing, "Competitions among service providers in cloud computing: A new economic model," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 2, pp. 866–877, Jun. 2018.
[16] S. Alneyadi, E. Sithirasenan, and V. Muthukkumarasamy, "Adaptable N-Gram classification model for data leakage prevention," in *Proc. 7th Int. Conf. Signal Process. Commun. Syst. (ICSPCS)*, Dec. 2013, pp. 1–8.
[17] G. Giannakopoulos, V. Karkaletsis, G. Vouros, and P. Stamatopoulos, "Summarization system evaluation revisited: N-Gram graphs," *ACM Trans. Speech Language Process.*, vol. 5, no. 3, 2008, Art. no. 5.
[18] Y. Lu, X. Huang, Y. Ma, and M. Ma, "A weighted context graph model for fast data leak detection," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2018, pp. 1–6.
[19] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1548–1560, Aug. 2011.
[20] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-CALD-02-107, 2002.
[21] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 1, pp. 55–67, Jan. 2008.
[22] D. D. Lewis, *Reuters Dataset*. Accessed: Sep. 26, 2017. [Online]. Available: http://www.daviddlewis.com/resources/testcollections/

XIAOHONG HUANG received the B.E. degree from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2000, and the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2005. Since 2005, she has been with BUPT, where she is currently an Associate Professor and the Director of the Network and Information Center, Institute of Network Technology. She has authored over 50 academic papers in the areas of wavelength division multiplexing optical networks, IP networks, and other related fields. Her current interests are Internet architecture, software-defined networking, and network function virtualization.



YUNLONG LU received the B.E. degree in electronic information science and technology from Beijing Forestry University in 2012 and the master's degree in computer technology from the Beijing University of Posts and Telecommunications, Beijing, China, in 2015, where he is currently pursuing the Ph.D. degree with the Institute of Network Technology. His research interests lie on the computer network security and data privacy.

DANDAN LI received the B.E. degree from Zhengzhou University, Zhengzhou, China, in 2013, and the Ph.D. degree from the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2017. She is currently a Lecturer with BUPT. She has authored over 10 academic papers in the *Physical Review Series*, the *IET Information Security*, and other related fields. Her current interests are sequence cryptography, quantum cryptography, and network security.

MAODE MA received the Ph.D. degree in computer science from The Hong Kong University of Science and Technology in 1999. He is currently an Associate Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. He has authored or co-authored over 300 international academic publications, including over 140 journal papers and over 160 conference papers. He has extensive research interests including network security and wireless networking. He is a fellow of IET, a Senior Member of the IEEE Communication Society and the IEEE Education Society, and a member of ACM. He currently serves as the Editor-in-Chief for the *International Journal of Computer and Communication Engineering* and the *International Journal of Electronic Transport*. He also serves as a senior editor or an associate editor for other five international academic journals. He is the Chair of the IEEE Education Society, Singapore Chapter, and the ACM, Singapore Chapter. He is serving as an IEEE Communication Society Distinguished Lecturer.