A Weighted Context Graph Model for Fast Data Leak Detection

Yunlong Lu, Xiaohong Huang, Yan Ma
Institute of Network Technology
Beijing University of Posts and Telecommunications
Beijing, China
{luyunlong, huangxh, mayan }@bupt.edu.cn

Maode Ma
School of Electrical and Electronic Engineering
Nanyang Technological University
Singapore
emdma@ntu.edu.sg

Abstract—**Data leakage prevention (DLP) uses a series of techniques to detect and prevent the sensitive data leakage caused by insider threat. Current detection methods either fail to achieve high accuracy toward transformed data or fail to reduce computational complexity. To ensure high detection accuracy and reduce computational complexity, we propose a Weighted Context Graph Model (WCGM) in this paper. The main goal of WCGM is three folds. First, the weighted context graph is proposed to build the contextual relation of data, based on which sub-graph matching method is used to calculate similarity features between tested data and pre-defined template. Second, machine learning algorithms are used to classify the tested data based on the similarity features of its context graphs. Third, privacy-preserving graph masking method is proposed to protect the data privacy of data holders. Extensive simulation results show that the proposed WCGM is able to achieve significant enhancement in terms of running time and accuracy.**

Keywords—data leak; weighted context graph; graph matching; machine learning algorithms

I. INTRODUCTION

With the arrival of Big Data era, data has penetrated into every field of our work and life, bringing people more and more benefit and convenience. But at the same time, data security has become a much more serious problem. In recent years, sensitive data such as the government confidential documents, corporate business secrets, private personal information and others have been leaked from time to time. For example, WikiLeaks leaked U.S. diplomatic cables in 2010, while Edward Snowden leaked the senior confidential documents of National Security Agency (NSA) in 2013 and in 2016 Yahoo disclosed 1 billion user account data. These incidents have brought serious damage to the data holders and the communities.

As sensitive data leakage has attracted more and more attention recently, data leak detection becomes particularly essential to protect data from unwilling disclosure to the unauthorized entities, and to ensure the privacy of the sensitive data. Academic researchers have paid much more attention on the Data Leak Detection (DLD) recently, especially data in the state of transmission. Existing studies mainly consist of two categories: content-based methods and context-based methods.

This work has been supported by Project U1603261 supported by Joint Funds of National Natural Science Foundation of China and Xinjiang.

The content-based approaches include predefined rules (e.g. regular expression), fingerprinting, statistical analysis and machine learning classification. Most of the traditional content-based methods use predefined regular expressions to perform precise detection [1]. The fingerprinting method is one of the traditional content-based methods to calculate the hash value of the file for comparison. But it can't deal with the transformed data because the hash value will change a lot if the original data makes any little change. To make up for this weakness, some improved methods, use the blocking hash instead of the overall hash to tolerate a certain text transformation [2,3]. The statistical analysis approach is mainly based on the term frequency characteristics of the text. The n-gram classification method calculates the offset characteristics of the detected file from the template based on the frequencies of words, to distinguish whether there is a data leak occurred [4]. The machine learning classification algorithms such as naïve Bayes [5], support vector machines (SVM) [6,7] are also widely used to identify the categories of the files. But the machine learning classifiers usually learn the most common statistical features, while most of the training files are non-sensitive, thus the most significant statistical features are mainly non-sensitive features instead of sensitive features, making it hard to identify the sensitive data from the non-sensitive. Moreover, the content-based approaches mentioned above take little contextual associations and document structures into consideration. As a result, the detection can't match the related dynamic sensitive data very well.

The context-based methods are much relatively new compared to the content-based methods in data leak detection. Recent studies attempt to perform the detection by adding the context information in matching. The N-gram method, widely used in text analysis, is also applied in data leak prevention [8]. There are other improved complex ways to build N-grams besides by the adjacent words [9], and the improved N-gram method is also proposed by using subsequence-preserving sampling [10]. But most N-gram based methods in data leak detection only consider very simple semantic information, for the complexity will increase rapidly with the increase of N (usually take 2 or 3). To address the above problem, the Term Frequency - Inverse Document Frequency (TF-IDF) is used to perform the detection [11]. Some studies have already tried to use graphs to represent the text instead of word vectors used in machine learning, to better reflect the text semantic [12,13]. The CoBAn [14] clusters the document based on the graphs of confidential terms and context terms to detect data leakage, achieving good performance. But the context terms it used also bring much calculating complexity in matching. Moreover, it calculates the confidential score based on the confidential and context terms, taking less direct links between confidential terms into consideration. In summary, the semantic of sensitive data is different from that in Natural Language Processing (NLP), the weight of sensitive key terms is much more important and the relations between the terms could be more simplified to describe the sensitive semantic directly. What's more, most of the context-based methods need to provide the original text to the detection agent to obtain the context information. For the original data is sensitive, the privacy of data may be not protected well at the agent's side (the agent may be attacked or controlled by malicious users to learn the sensitive data).

In this paper, a new model named Weighted Context Graph Model (WCGM) is proposed to achieve high detection accuracy with low computational complexity. The main contributions of this paper can be summarized as follows:

- The computational complexity is reduced by using the proposed weighted context graph to build the contextual relation of data, based on which sub-graph matching method is used to calculate similarity features between tested data and pre-defined template.
- To better tolerate and match the transformed data, machine learning algorithms are used to classify the tested data based on its context graph's similarity features for fast detection.
- Privacy-preserving Graph Masking method is applied to the key sensitive terms by using irreversible encryption, keeping the sensitive content masked, while the sematic correlations are not affected, which can protect the data privacy of data holders.

Experiments have been done to evaluate the performance of the proposed method. The results show that the proposed method achieves better performance in terms of detection accuracy, recall rate and running time, especially the running time.

The rest of this paper is organized as follows. In Section II, the weighted context graph model is introduced. In Section III, the maximum sub-graph matching and detection are discussed in detail. Then, the evaluation results are presented in Section IV. Finally, the paper is summarized in Section V.

II. Weighted Context Graph Model

In this section, the problem definition of this paper is introduced at first, describing the overview application scenario of data leak detection in this paper. The proposed WCGM consists of three phases: sensitive weighted context graph construction, privacy-preserving graph masking, maximum sub-graph matching. The sensitive context graph construction and the privacy-preserving graph masking are performed on the side of the data holder, while the maximum sub-graph matching runs on the detection agent's side. The details of these phases are described in the following subsections.

A. Problem Definition and Solution Overview

We focus on the scenario that the local area network reveals data to the external network for reasons such as operational errors or human errors or malicious attacks. It is a serious threat from the insiders to leak the sensitive data to the outsiders. There are mainly two participants in the detection model: the data holder and the detection proxy. The detection proxy is deployed at the network's export side as an agent. The proxy inspects the outgoing content to identify the data leakage. The overall scenario is shown in Fig. 1.

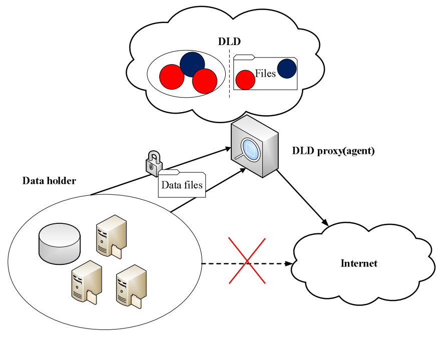

Fig. 1. The overall application scenario of DLD

The sensitive data in this paper is assumed in the form of sensitive files, while the other data is defined as non-sensitive files. Given a collection of data files *D*, which consists of two categories: sensitive data and non-sensitive data, with *C* = {sensitive, non-sensitive} to represent. The goal of this paper is to build the mapping $\Phi: D \to C$ by supervised method, then further build the function $\Phi': S \to C$ to classify the outgoing files S of the local network into the approximate categories, thus it is possible to detect whether the sensitive data has been leaked.

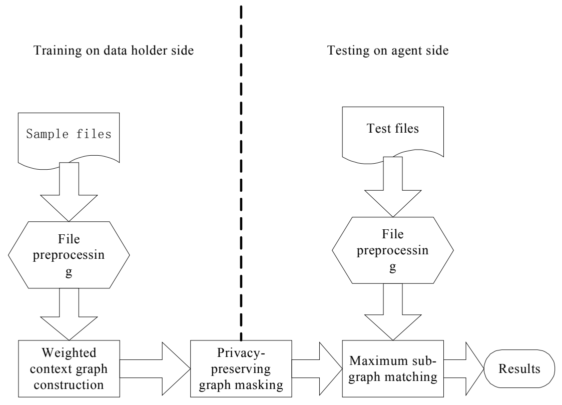
Training on data holder side
Testing on agent side
Sample files
Test files
File preprocessing
File preprocessing
Weighted context graph construction
Privacy-preserving graph masking
Maximum sub-graph matching
Results

Fig. 2. The overview of WCGM

This paper also considers the proxy as semi-honest, which may try to learn the sensitive data of the data holder for some reasons (commercial interests or malicious attacks). In this situation, the data holder may want to perform the detection without revealing any plain sensitive data to the agent.

The WCGM is proposed to solve the problems mentioned above. Specially, WCGM uses graph masking to deal with the situation of semi-honest proxy. As mentioned above, WCGM consists of three modules. The overview is described in Fig. 2

B. Sensitive Weighted Context Graph Construction

The weighted context graph is defined as follows:

Definition 1 [Weighted Context Graphs]: A weighted context graph is defined as $G = \{N, E\}$, where *N* is the set of nodes (key terms), and *E* is the set of weighted edges (an edge represents the association between two nodes). A weighted context graph is a summary of the text, which preserves the main keywords by the nodes and the context by the edges. Each node has a value of word text and a weight W_N, and each edge also has a weight W_E.

The weighted context graphs are built locally on the data holder's side. The data holder provides files of two categories: sensitive and non-sensitive. The goal of building the weighted context graph is to keep the key feature information of the text as much as possible. It is necessary to be streamlined, but also to maximize the retention of the characteristics of the text, so as to achieve a balance between the efficiency and the accuracy.

The features of sensitive data are mainly concerned with the key sensitive terms and their correlations. Extracting the key terms from the sensitive files is a key step to construct the context graph. For the frequency of words is an important characteristic in each file, the TF-IDF is widely used in representing the features of a file. This paper has adjusted the TF-IDF method to extract the sensitive key words in sensitive files. Firstly, each document is processed to remove the stop words and any meaningless words like 'the' and 'ing' suffix. The processed documents will be split into grams of different length by word segmentation. And then for each term in a document, the adjusted TF-IDF will be calculated. Let d_j^S denotes the *j*-th segmented file in sensitive files set d^S, $n_{i,j}$ denotes the number of occurrences of the *i*-th gram t_i in d_j^S, the TF for t_i is $tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$, where the denominator is the sum number of all terms in d_j^S. The IDF is $idf_i = log\frac{|d^S|}{|\{j: t_i \in d_j\}|}$, where $|d^S|$ is the total number of sensitive files, $|\{j: t_i \in \ d_j\}|$ is the number of non-sensitive documents that contain term t_i. Thus the TF-IDF is $tf\mathrm{idf}_{i,j} = tf_{i,j} \times idf_i$. We choose the key grams according to the adjusted TF-IDF value, and the key grams are the nodes of the context graph for sensitive file d_j^S.

For each node *i* of the graph, it has two properties, value and weight. The value is the text content of the word grams. The weight $W_i = \frac{tdf_i}{\sum tdf_i}$, where tdf_i denotes its adjusted TF-IDF value, $\sum tdf_i$ denotes the sum of adjusted TF-IDF value of all nodes in file d_j^S. For any two nodes, if the distance between them is less than N, then an edge is set between them. Each edge has a property of weight reflecting the distance between two nodes, the *i*-th edge's weight $W_{ei} = \frac{N-n_i}{N}$, where n_i is the number of terms between the two nodes in the file d_j^S after segmentation, and N is the maximum threshold of length for an edge connection. The closer two nodes are, the higher W_{ei} is. The overview of a weighted context graph is shown in Fig. 3.

The whole sensitive weighted context graph G_T is derived from each document graph G_{Ti} . We update each G_{Ti} to the G_T by adding and merging the union of nodes and edges to G_T .

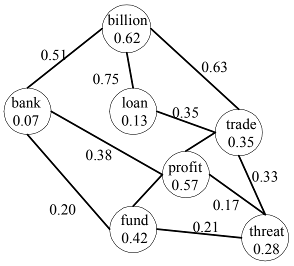

Fig. 3. Example of a weighted context graph

We use matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

to describe the context between the grams, that is, the edges and their weights of the context graph. Rows of the matrix denote the ranked word grams of the context graph, and columns denote the same ranked grams, which are also the nodes of the context graph. The a_{ij} means the edge between the *i*-th and *j*-th node, $a_{i,j} = \{0, w, -1\}$, where 0 denotes the two nodes are the same one, -1 denotes there is no edge between the two nodes, and w denotes the weight of the edge, $w > 0$.

Algorithm 1: Sensitive Weighted Context Graph

```
Input: documents set $D$, max threshold $N$
Output: graph $G_T$ - nodes set and edges matrix $A$
1: Initialization: set all weight $W$ = 0.0, read data from $D$
2: Run word segmentation, remove the relevant auxiliary
   words,  $D \to D^{seg}$
3: while $D_i^{seg}$ in $D^{seg}$ do
4:     Compute TF-IDF in $D_i^{seg}$, choose the key sensitive terms
       as nodes, compute their edge matrix $A$, get $G_{Ti}$
5: end while
6: Merge the $G_{Ti} \to G_T$: compute the merged weight $W$ and
   edge matrix $A$
7: end
```

C. Privacy-preserving Graph Masking

As the data holder's network size may be too large or too complex to carry out the detection himself, the WCGM is provided as a third-party service. However, the DLD proxy may be semi-honest, for it may be attacked or controlled by malicious users. For the data holder needs to provide his own sensitive data to the agent, the data holder is exposed to great risks once the agent turns malicious.

Therefore, it is essential to prevent the sensitive data from being learned by the agent in its detection phase. To figure this problem out, a privacy-preserving graph masking method is proposed in this section. Firstly, the data holder constructs the context graph locally as described in section B to obtain the sensitive weighted context graph. The graph consists of nodes set and edges matrix. Then the data holder will apply the irreversible encryption to the value of each node in the context graph, while the node weights and edge correlations are preserved. The masked sensitive context graphs, revealing nothing about the text content, will be provided to the detection proxy as profiles. The overall profile of the method is shown as Fig. 4:

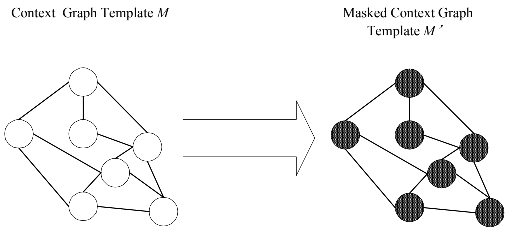
Context Graph Template *M*
Masked Context Graph Template *M ′*

Fig. 4. The masking process of context graph template

The advantage of the proposed privacy-preserving graph masking method is that it can preserve the correlations between sensitive terms, and keep the values of sensitive terms unknown to the detection agent. It is precisely because of the characteristics of our proposed WCGM model that can perform the graph masking without breaking the sensitive correlations.

III. Context Graph Matching and Detection

A. Maximum Sub-Graph Matching

The files to be detected are classified into the correct categories (sensitive, non-sensitive) by matching the text graph G_{Ti} with the text graph template G_T , as shown in Fig. 5.

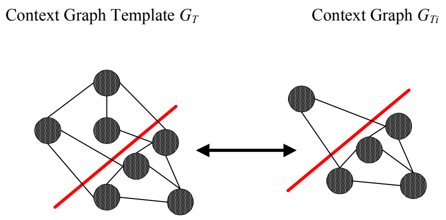

Fig. 5. Context graph matching

For the weighted context graph template G_T may be too large and the G_{Ti} is only a sub-graph of it, a sub-graph matching method named Maximum Sub-Graph Matching algorithm is proposed. There are some methods to calculate the similarity of two graphs [15,16]. To estimate the matching similarity between a template graph G_T and a file graph G_{Ti} , we mainly examine the intersecting degree of nodes and edges. The exact matching metrics are defined as follows.

- Node Similarity (NS): indicates the similarity weight of intersecting nodes between graph G_{Ti} and graph G_T . We first calculate the $G_{TI}(d) = G_T(s) \cap G_{Ti}(t)$, where $G_{TI}(d)$ denotes the intersecting nodes of G_T and G_{Ti} . The NS is defined as follows:

$$NS(G_{Ti}, G_T) = \sum_{n \in G_{TI}} W_n^{Ti} \times W_n^T \qquad (1)$$

- Edge Similarity (ES): The edge weight denotes the distance and importance of the edge. As before, we use matrix to store the matching similarity of edges. Rows and columns are nodes of $G_T(d)$, ranked according to their weights from large to small. One node is much more important if it has more links matched to the template, so we define the Node Correlation Rank (NCR) to denote the correlation importance of a node, each row has one common NCR in the matrix. The modified edge weight of G_T for e is:

$$\lambda(e, G_T) = \begin{cases} 0 & if\ e \notin E_T \\ w_e^{G_T} & if\ e \in E_T \end{cases} \qquad (2)$$

where e denotes the edge in G_{Ti} , $w_e^{G_T}$ denotes the edge weight in graph template G_T.

$$ES(E_{ti}, E_T) = \sum_{a_{ij} \in A, i>j} \lambda(e_{ij}, G_T) a_{ij} \mu(i,j) \qquad (3)$$

where a_{ij} is the element of the matrix as mentioned before. $\mu(i,j) = \log(NCR/2)$ denotes the crossing correlation degree of the edge, where $\mathrm{NCR} = \left|\{a_{ij}: a_{ij} > 0, i = i, 0 \le j < i\}\right|$ is the number of positive elements in row i.

- Edge containment similarity (ECS): ECS indicates the proportion of intersecting edges of G_{Ti} and G_T in their union edges of $G_{TI}(d)$.

$$ECS(G_{Ti}, G_T) = \frac{\sum_{e \in G_{Ti} \cap G_T} (w_e^{Ti} + w_e^T)}{\sum_{e \in G_{Ti} \cup G_T} (w_e^{Ti} + w_e^T)} \qquad (4)$$

where $w_e^{Ti} = 0$, if e only exists in G_T with w_e^T, vice versa. The numerator denotes the intersecting edges that have two same nodes, the denominator denotes all the edges sharing at least one same node.

For each single dataset, the NS, ES and ECS of each file from the training set are calculated first. Then a variety of machine learning algorithms are tested with parameters of the training set, such as Naïve Bayes, Bayes Net, Random Tree, Random Forest, to determine which has better performance to construct the classifier. In the detection phase the constructed classifier will be used to classify the corresponding test set.

Algorithm 2: Graph Matching and Detection

Input: Context graph G_{Ti}^{train} , G_{Ti}^{test} , G_T

Output: The sensitivity of G_{Ti}^{test}

```
1: while $G_{Ti}^{train}(i)$ in $G_{Ti}^{train}$ do
2:    Calculate the NS, ES and ECS between $G_{Ti}^{train}(i)$ and $G_T$
3: end while
4: Test classification algorithms on the set with the three
   parameters
5: Select the best algorithm to construct the classifier
6: Repeat step 1-3 on $G_{Ti}^{test}$
7: Test with the corresponding classifier
8: end
```

B. Complexity of WCGM

The WCGM mainly consists of two phases – weighted context graph construction and graph matching detection. For the weighted context graph construction is performed at the data holder's side asynchronously and offline, the complexity of detection phase is much more important, which is applied in real-time. To detect whether a file is sensitive, there are mainly three parts: (a) constructing the context graph of the file; (b) calculating its NS, ES and ECS; (c) classifying to decide the sensitivity. For one detected file, characterized with three parameters, once the model of classification is established, the complexity of phase (c) is too small to count. Assuming there are C terms in the document, the count of the file's key terms is M, the size of the window to establish an edge between two nodes is N, the average edges of one node is P, usually $P < N$. Thus, the complexity of (a) and (b) are as follows:

- Constructing context graph: Calculating all terms weight needs to read the file once, to record the count and position of each term in the document. The complexity of this process is $O(C)$. Then adding the correlation edges needs to find the adjacent key terms in the window of each key term, the complexity is $O(MN)$. The overall complexity of this stage is $O(MN + C)$. Usually $MN > C$, thus the complexity can be reduced to $O(MN)$.
- Calculating NS, ES, ECS: The hashmap is used to store the context graph, whose time complexity of lookup is $O(1)$. For each key term the cost to calculate NS is $O(1)$, to calculate ES is $O(P)$, ECS is $O(2P)$, thus the overall complexity is $O(M(2P + P + 1))$, which can be reduced to $O(MP)$.

In summary, the overall complexity of the above steps is $O(MN + MP)$, since both N and P are constants, and $P < N$, the complexity can be reduced to $O(MN)$. The complexity is liner to the number of key terms in the file and the size of edge window, while the two parameters are both constant values in our evaluation. Therefore, it can be concluded that the WCGM is capable of being performed in real time.

IV. Evaluation

In this section, a series of experiments have been done to evaluate the performance of the proposed detection model. A series of articles appeared on Reuters newswire, have been chosen as the basic data set. The documents in it have been assembled and indexed with categories, each category divided into a training set and a test set. For many categories in the dataset are too small or too peculiar to perform further evaluation, the data files from 23 different categories, whose numbers of documents are thousands, are screened out from the basic dataset. For there are very few published sensitive data files, the sensitive data is chosen from the basic dataset by assuming one category as sensitive and the others are not each time. The statistical experiments are performed on the 23 data sets, the edges threshold N takes 20 by experimental experience.

In the training phase, the overall template graph is constructed and the NS, ES, ECS of each single document towards the template are calculated. Then a variety of machine learning algorithms including Naïve Bayes, Bayes Net, Random Tree, Random Forest, have been tested to determine which one suits the training set best, and to construct the classifier with the parameters NS, ES, ECS. The tests are performed on the test sets corresponding to the training sets, using the constructed classifier. The performance of CoBAn, an advanced method superior to most methods, proposed in [14], is also compared with our model on the same dataset.

Different datasets with various number of files have been tested. The accuracy of ordinate in Fig. 6 denotes the rate of correctly classified cases. The recall given in Fig. 7 denotes the rate of sensitive files also classified as sensitive, describing the capability to detect the leak of sensitive data. As is denoted WCGM achieves better performance in most results, especially in terms of recall. The reason is that the files in the training set and test set of one category are quite different, the WCGM focus on the correlations of key terms by using edges instead of the common context terms in CoBAn, which is more appropriate to the transformed data. The fluctuation trends of the accuracy and recall curves are relatively consistent, indicating that the fluctuations are related to the characteristics of the tested datasets (e.g., some categories of data may be too close to distinguish out).

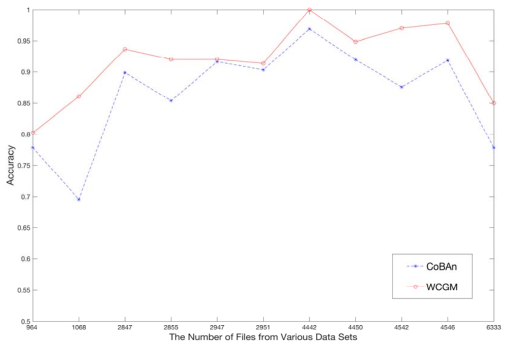

Fig. 6. The accuracy of CoBAn and WCGM

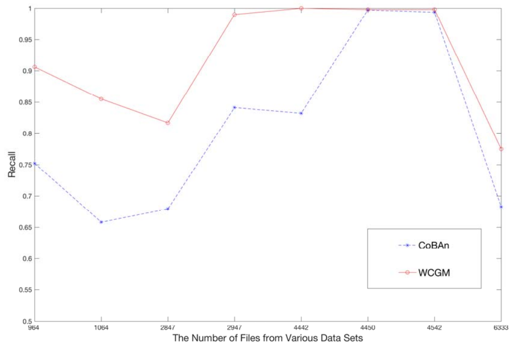

Fig. 7. The recall of CoBAn and WCGM

The total running time and detection time (in seconds) of CoBAn and WCGM are also given in Fig. 8. The running time results of WCGM are much better. Moreover, the running time of WCGM increases slowly as the number of files increases, while CoBAn increases much. That is because the complexity of WCGM is $O(MN)$ as mentioned in Section III, which is liner to the number of key terms in the file and the size of edge window, not much related to the file size. While the complexity of CoBAn is $O(TC)$ [14], which is liner to the number of the terms in the file and the number of clusters. The receiver operating characteristic (ROC) curves are also provided in Fig. 9. The TP axis denotes the detection rate of real sensitive cases, while FP denotes the "false alarms".

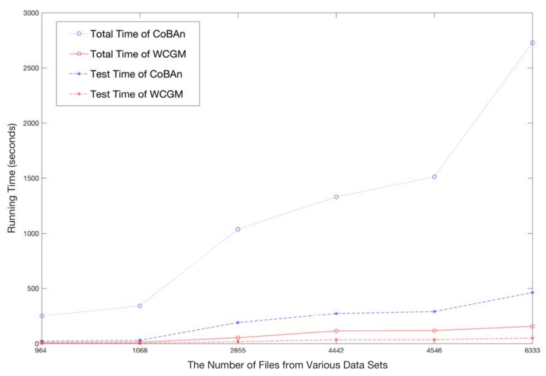

Fig. 8. The running time of CoBAn and WCGM

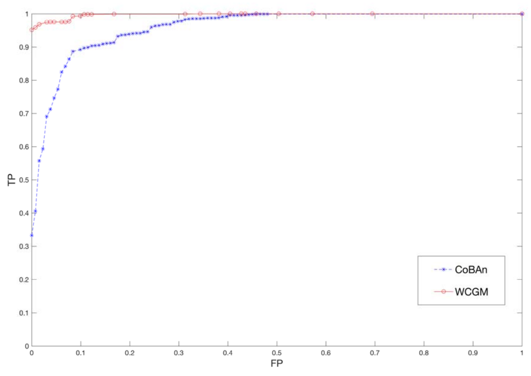

Fig. 9. The ROC curves of CoBAn and WCGM on a dataset with 4442 Files

As is shown above, our proposed model WCGM performs better on the same data set in terms of detection accuracy, recall, and especially running time. Moreover, the training sets and the test sets used in this experiment are quite different files only in the same topic category. It indicates that the proposed method has good data compatibility to detect diverse transformed data.

V. Conclusions

In this paper, we presented a Weighted Context Graph Model (WCGM) for fast data leak detection. It reduces the complexity by using the key sensitive terms as nodes, and the contextual associations as edges to construct the weighted context graph. To better match the transformed data, the proposed maximum sub-graph matching method and machine learning algorithms are used to estimate the similarity between the tested data and the template, and to determine the sensitivity of the tested data. In addition, the privacy of the sensitive data has been protected towards the proxy by masking the content of the template graph's nodes. Evaluation results reveal that the proposed model achieves better performance in accuracy and recall, and especially in terms of running time, illustrating that it can perform the fast data leak detection in real time.

References

[1] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. Turner, "Algorithms to accelerate multiple regular expressions matching for deep packet inspection," in ACM SIGCOMM Computer Communication Review, vol. 36, no. 4. ACM, 2006, pp. 339–350.

[2] X. Shu and D. D. Yao, "Data leak detection as a service," in Security and Privacy in Communication Networks, ed: Springer, 2012, pp. 222-240.

[3] X. Shu, D. Yao, and E. Bertino, "Privacy-preserving detection of sensitive data exposure," IEEE transactions on information forensics and security, vol. 10, no. 5, pp. 1092–1103, 2015.

[4] S. Alneyadi, E. Sithirasenan, and V. Muthukkumarasamy, "Word N-gram based classification for data leakage prevention," in 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 2013, pp. 578-585.

[5] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, and C. D. Spyropoulos, "An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages," in Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, 2000, pp. 160-167.

[6] H. Drucker, D. Wu, and V. N. Vapnik, "Support vector machines for spam categorization," IEEE Transactions on Neural networks, vol. 10, no. 5, pp. 1048–1054, 1999.

[7] M. Hart, P. Manadhata, and R. Johnson, "Text classification for data loss prevention," in International Symposium on Privacy Enhancing Technologies Symposium, 2011, pp. 18-37.

[8] S. Alneyadi, E. Sithirasenan, and V. Muthukkumarasamy, "Adaptable n-gram classification model for data leakage prevention," in Signal Processing and Communication Systems (ICSPCS), 2013 7th International Conference on, 2013, pp. 1-8.

[9] G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh, and L. Chanona-Hernández, "Syntactic N-grams as machine learning features for natural language processing," Expert Systems with Applications, vol. 41, pp. 853-860, 2014.

[10] X. Shu, J. Zhang, D. D. Yao, and W.-C. Feng, "Fast detection of transformed data leaks," IEEE Transactions on Information Forensics and Security, vol. 11, no. 3, pp. 528–542, 2016.

[11] S. Alneyadi, E. Sithirasenan, and V. Muthukkumarasamy, "Detecting Data Semantic: A Data Leakage Prevention Approach," in Trustcom/BigDataSE/ISPA, 2015 IEEE, pp. 910-917.

[12] G. Giannakopoulos, V. Karkaletsis, G. Vouros, and P. Stamatopoulos, "Summarization system evaluation revisited: N-gram graphs," ACM Transactions on Speech and Language Processing (TSLP), vol. 5, no. 3, p. 5, 2008.

[13] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 8, pp. 1548– 1560, 2011.

[14] G. Katz, Y. Elovici, and B. Shapira, "CoBAn: A context based model for data leakage prevention," Information Sciences, vol. 262, pp. 137-158, 2014.

[15] T. A. Schieber, L. Carpi, A. D́ıaz-Guilera, P. M. Pardalos, C. Masoller, and M. G. Ravetti, "Quantification of network structural dissimilarities," Nature communications, vol. 8, p. 13928, 2017.

[16] S. Soundarajan, T. Eliassi-Rad, and B. Gallagher, "A guide to selecting a network similarity method," in Proceedings of the 2014 SIAM International Conference on Data Mining. SIAM, 2014, pp. 1037–1045.