CloudDLP: Transparent and Automatic Data Sanitization for Browser-based Cloud Storage

Chuanyi Liu*§, Peiyi Han†*, Yingfei Dong‡, Hezhong Pan†*, Shaoming Duan*, Binxing Fang*
*Harbin Institute of Technology (Shenzhen), Shenzhen, China
†Beijing University of Posts and Telecommunications, Beijing, China
‡University of Hawaii, Hawaii, USA
§Correspondence to: cy-liu04@mails.tsinghua.edu.cn

***Abstract*—Because cloud storage services have been broadly used in enterprises for online sharing and collaboration, sensitive information in images or documents may be easily leaked outside the trust enterprise on-premises due to such cloud services. Existing solutions to this problem have not fully explored the tradeoffs among application performance, service scalability, and user data privacy. Therefore, we propose CloudDLP, a generic approach for enterprises to automatically sanitize sensitive data in images and documents in browser-based cloud storage. To the best of our knowledge, CloudDLP is the first system that automatically and transparently detects and sanitizes both sensitive images and textual documents without compromising user experience or application functionality on browser-based cloud storage. To prevent sensitive information escaping from on-premises, CloudDLP utilizes deep learning methods to detect sensitive information in both images and textual documents. We have evaluated the proposed method on a number of typical cloud applications. Our experimental results show that it can achieve transparent and automatic data sanitization on the cloud storage services with relatively low overheads, while preserving most application functionalities.**

***Index Terms*—data loss prevention, data sanitization, data security, cloud storage**

I. Introduction

Nowadays, cloud storage services or cloud applications with file storage functionality, such as Dropbox, Box, and Salesforce, etc., are increasingly used in enterprises for online sharing and collaboration. However, sensitive information may be easily (accidentally or maliciously) shared outside of the trust premises due to these cloud services. In June 2014, Google patched a flaw in Google Drive that may grant unauthorized parties access to a subset of shared documents under certain circumstances. Dropbox and Box were also affected by similar security issues, where major vulnerabilities allowed third parties to discover private file transfer links. Besides, researchers have identified several defects or misuses in Amazon Simple Storage Service (S3) that may leak sensitive information such as trade data [1], military secrets [2], private medical data, affecting over 150 thousand people [3] and about 198 million American voter records [4]. Therefore, it is critical to ensure that all sensitive information can be properly detected, protected, or sanitized, before uploading data from on-premises to cloud storage.

To protect against data leakage in the browser-based cloud storage, various approaches and systems have been introduced. Data Loss Prevention (DLP) technology is critical in this area. DLP software products such as SPIRION [5], CUSpider [6], and CipherCloud [7], detect sensitive information using massive regular expressions, custom keywords, or domain specific entity corpora. Unfortunately, the detection precision is still low even though traditional machine learning algorithms are integrated into these systems. To address this problem, existing solutions mostly focused on textual data rather than images and complicated documents. For example, ShadowCrypt [8] only supports encrypting cleartexts but not able to encrypt complicated files. M-Aegis [9] only supports encryption for textual data. MessageGuard [10] creates a file upload overlay using HTML iFrames to protect data while sacrificing the storage capacity and search functionality. Although Cloud Access Security Broker (CASB) [11] solutions (adopted by commercial companies like Skyhigh Networks [12] and CipherCloud [7]) provide protection of sensitive information in files, they have to adapt various services by reverse engineering service-specific protocols, which is time-consuming and labor-intensive (as further discussed in Sec. II). In fact, it is very difficult to simultaneously achieve good privacy and usability levels after encryption. Making it even worse, these solutions often degrade the scalability of various applications in practice.

In this paper, we introduce CloudDLP, the first system that automatically and transparently detects and sanitizes both sensitive image and textual information, without compromising user experience or application functionality on browser-based cloud storage. Additionally, CloudDLP is designed to provide scalability for various cloud applications. It can automatically identify and capture files without requiring cooperation from cloud providers or complicated protocol reverse engineering. Our system instrument JavaScript snippets to the web application pages with the support of gateways. The injected JavaScript snippets are used to identify file requests. Then CloudDLP will intercept, detect, and redact or sanitize sensitive data, especially images and documents which contain a wide range of private information that may be shared unintentionally. To prevent sensitive information escaping from on-premises, CloudDLP detects sensitive information from images and textual documents using deep learning methods. Moreover, privacy-utility trade-offs can be achieved by automatic sanitization in CloudDLP. Compared to traditional rule-based approaches, our automatic detection and sanitization methods based on deep learning methods have obvious advantages and show superior performance. With CloudDLP, enterprises can effectively prevent core data leakage, and can also ensure sensitive data security and regulatory compliance in the cloud.

To validate the feasibility of the proposed method in practice, We have implemented and evaluated CloudDLP in ten real-time browser-based cloud storage services, including email, storage, and office software. The experiments on these applications successfully demonstrate the effectiveness and generality of the proposed approach. The performance evaluation shows that CloudDLP is cable to protect enterprise sensitive data with fairly low overheads.

Our contributions can be summarized as follows:

- We propose CloudDLP, the first system that provides automatic and transparent sensitivity detection and sanitization in various browser-based cloud storage. It is easy to adapt to new applications with minimal efforts.
- To preserve the utility of the application while preserving privacy, our system detects sensitivity in images and textual documents, and performs automatic sanitization through deep learning methods.
- We have implemented and tested CloudDLP with various popular cloud applications, such as Gmail, Box, Dropbox, Salesforce, etc. Our experimental results demonstrate that CloudDLP is effective in real-time applications with fairly low overheads.

The rest of the paper is organized as follows. We review the related work in Section II. We present the overall system architecture and implementation Section III. We present performance evaluation and case studies in Section IV. Finally, we conclude this paper in Section V.

II. Related Work

A number of solutions to protect sensitive data for cloud storage services have been proposed. In this section, we discuss the advantages and disadvantages of these solutions.

Data Loss Prevention (DLP) system is used to protect three types of data: *the data is at rest, data is in motion, and data is in use* [13]. In this paper, we focus on data in motion. Several non-automated methods are based on regular expressions, such as SPIRION [5], CUSpider [6], and CipherCloud [7]. They detect sensitive information based on pre-defined keywords and domain-specific entities through regular expressions. The detection precision of these systems are often low due to many false positives, and they also require large-scale dictionaries. Several solutions have adopted machine-learning based methods for aumatically detection, but their performance is fairly disappointing. Elisa Costante et al [14] proposed a DLP framework combined signature-based and anomaly-based methods, which enable detection and prevention simultaneously. First, this framework trains a model to recognize user behaviors using machine learning algorithms. Then building and updating the attack signatures automatically according to operation feedbacks on alerts. However, the accuracy of these automatically approaches has been lower than some manually specified methods. Gomez-Hidalg et al. [15] presented an automatic approach using NER (Named Entity Recognition). They used a dataset extracted from Twitter using Google Insight [16]. Unfortunately, this approach cannot be extended to other data types (such as files and images), and only works for the textual data. Ong Y J et al [17] introduced a system that can detect sensitive information on various degrees of granularity within the document by using machine learning and deep learning algorithms. Their deep-learning based detection models use semantic context information about documents to predict whether they contain sensitive information. This system effectively detects sensitive data from the documents and help users mitigate future threats. However, it also shows shortcomings and limitations. First, this system cannot detect sensitive information in images and other non-document data. Second, it is only applied to a specific domain that cannot be applied to the general domain.

Cloud Access Security Broker (CASB) [11] is a proxy-based approach to protect user privacy. Many companies such as CipherCloud [7] and Skyhigh Network [12] launched their own CASB products. CASB is deployed between a user and an application. When the proxy intercepts sensitive data through protocol analysis, it encrypts the data and sends to the application. However, users need to know the protocol of each application in advance; when the application protocol changes, it may not work. Therefore, this approach is difficult to apply to every cloud application.

Other works. ShadowCrypt [8] runs as a browser extension and encrypts data before the application code accesses it. It replaces input elements in a page with secure, isolated shadow inputs and replaces cleartext with the encrypted text. However, ShadowCrypt is unable to achieve encryption for non-textual data such as complicated documents, images, etc., and can not support any mobile applications. CryptDB [18] encrypts user confidential data between an application server and a database server. CryptDB includes a proxy that interposes user operations and translates normal queries into queries on encrypted data. CryptDB effectively protects confidential data against the database administrator and allows the user to query transparently on encrypted data. However, it only protects the database and does not support cloud applications. Mylar [19] is an encryption system which using Meteor JavaScript framework. Users encrypt their data by programming in the Meteor JavaScript framework. However, Mylar not only suffers from the lack of compatibility, but also cannot perform data analysis in a cloud environment. Virtru [20] is an email encryption system used to protect webmail contents of being leaked. Although it can protect webmail data very well, it cannot be extended to other non-mail web applications and also only work for a few mail providers. These solutions are only used for specific domains.

III. System Architecture

A. Design Goals

In this paper, we aim to develop methods to protect sensitive information of images and documents from leaking out of

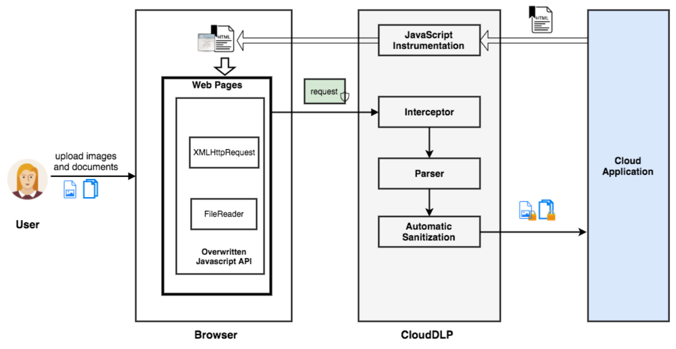


Fig. 1. System Architecture of CloudDLP.

enterprise on-premises to insecure cloud storage via *browser-based interfaces*. Developing a privacy preservation approach for browser-based cloud storage services is usually not an easy task, due to the following requirements:

1. **Security.** The solution need ensure that sensitive data is protected before leaving enterprise on-premises to the cloud.
2. **Usability.** The method need better preserve user experiences and application functionalities, such as document previewing and editing.
3. **Scalability.** The solution must be easy to maintain and be highly scalable for various applications.

B. Threat Model

CloudDLP should be deployed at the edge of an enterprise network, where main security restricted policies can keep most attackers out. We assume that an internal enterprise network is security and trustworthy. But cloud storage service providers may be malicious. Sometimes, they may compromise the security of valuable user data due to commercial interests, legal requirements, or when they are compromised. Meanwhile, the client-side application code and the middleware on the network channel may also be exploited to exfiltrate sensitive information.

Additionally, we assume the operating systems, browsers, and network devices inside an enterprise firewall are trusted, which is a practical trusted zone in most organizations. CloudDLP does not provide protection against side-channel attacks.

C. CloudDLP Architecture

In this section, we present the architecture of CloudDLP. CloudDLP is an internet gateway deployed within the premises of an enterprise. A user interacts with the application as usual while CloudDLP seamlessly detects and redacts outgoing sensitive data. Fig. 1 shows the overall system architecture, which consists of several main components: (a) Interceptor, (b) Parser, (c) Automatic Sanitization, and (d) Packer. In the following, we will briefly describe the functionality of each component.

1) Interceptor: The interceptor is implemented as a secure proxy between enterprise users and cloud service providers. It intercepts network traffic such as HTTP/HTTPS from premises to the cloud. Furthermore, the interceptor can intercept the TLS connections with a certificate authorized by the enterprise via the “SSL man-in-the-middle” technique. Inspecting traffic is enforced to prevent sensitive user information being disclosed outside enterprises. It is also responsible for injecting the JavaScript snippets into the web pages of cloud applications. The JavaScript snippets are used to override the JavaScript native API such as XMLHttpRequest and FileReader. The snippets can intercept all the XHR requests, and then screen and identify file uploading requests.

Although third-party JavaScript libraries can provide various file and web access interfaces, many of them eventually need to invoke the primitive File and Web interface of JavaScript. Meanwhile, the File and XMLHttpRequest API Standard are fairly mature and have been rarely updated in recent years. So the implementation of CloudDLP can easily keep up with the current standard and has a low risk of out of sync with the current JavaScript File and Web API. This is a very low-frequency event compared with other development of cloud services. Therefore, we believe that CloudDLP is a practical solution both at present and in the near future.

2) Parser: The Parser is for application protocol parsing, file content extraction and document parsing, based on syntax and semantics analysis. It obtains data buffered for a user session and analyzes the request content format (including key-value, multi-part, etc.) to extract document fields.

3) Sanitizer: The Sanitizer has two main tasks: (a) detection of sensitive information within images and documents; (b) sensitive information hiding, in a way that the disclosure risk is minimized and, ideally, the utility of the sanitized image and text is maximized. We have developed several methods to detect sensitive information in images and textual documents through several advanced deep learning models. For example, we are using the scene text reading approach based on convolutional neural network (CNN) and recurrent neural network (RNN). CloudDLP associates the discovery of sensitive data in text to the recognition of Named Entities in Natural language processing (NLP). Thus private information in images can be redacted while preserving the utility.

4) Packer: The Packer module is able to assemble the sanitized images or documents and then to send them back for corresponding requests. At the end, the gateway can forward the protected request to the cloud.

As a concrete example, when an enterprise user visits a cloud storage service via CloudDLP. The interceptor injects the JavaScript Snippets into the requested web pages. Then, the snippets executed in the user's browser would hook the native JavaScript APIs including XMLHttpRequest and FileReader. It intercepts and labels all requests that include files. Algorithm 1 shows the CloudDLP workflow. CloudDLP first receives a connection from the browser of the user (line 1). The connection is always over HTTP or HTTPS. When the user interacts with the cloud application, the requests sent by the browser always include web pages. Then the interceptor of CloudDLP instruments the JavaScript snippets at the head of the web pages (line 3). The snippets can override the native JavaScript

Algorithm 1 The main CloudDLP algorithm

```
Input: B: the browser
Input: CSS: the Cloud Storage Service
Input: C: CloudDLP
1: while a connection c from B do
2:     if c.request contains html page then
3:         c.response ← $C.inject\_js\_snippet(c.response)$
4:     if c.request is identified as file uploading then
5:         file ← $C.parse(c.request)$
6:         file ← $C.sanitize(file)$
7:         c.request ← $C.pack(c.request, file)$
8:     send(c)
```

API such as XMLHttpRequest and FileReader. Thus, the snippets can intercept network traffic, capture file operations invoked by the application, and identify file requests. The tag label of each request indicates that this is a file request and the file content needs to be detected and sanitized. Then the parser will analyze the semantic of the request, extracts files from the connection and parses the documents (line 5). The sensitive information of the extracted files will be removed by the automatic sanitization (line 6). Finally, the packer rebuilds each file in associated requests and rewrites the sanitized file contents into the requests to ensure full usability and privacy of outgoing contents (line 7).

D. Automatic Identification of Files

Building a generic automatic approach for new and existing applications is one of the most significant design goals of CloudDLP. To achieve this goal, we need first recognize upload requests from a variety of applications at the proxy. In the following section, we first discuss two unsatisfactory approaches to identifying uploading requests, and then introduce the dynamic analysis of JavaScript and it use in FileCrypt.

One method is to only use regular expression matching for features extraction from uploading requests to test on whether it meets the uploading requirements. It calls for accumulated and sophisticated match rules derived from the analysis of all application protocols respectively. In addition, the matching rules in the proxy have to be adaptive once a cloud service protocol changes. Therefore, simply using regular expression matching has serious limitations. The second naive method is to use an inter-procedural string analysis [21], which can also extract an upload requests URL string, HTTP methods, and the relevant request data. However, this method usually leads to a low accuracy, since service suppliers normally compress and mix JavaScript codes. So inter-procedural string analysis cannot extract request information precisely.

In reality, file transmission in cloud applications is always accompanied by invoking of the XMLHttpRequest object. No contemporary browsers are lacking in a built-in XMLHttpRequest object, which plays a key role in defining a programming interface for data transfer between a web browser and a web server. Note that a file object here can be Blob [22], File, FormData [23], or ArrayBuffer [24]. Among these types, Blob and File refer to the constant file data object, while FormData stands for a set of key/value pairs revealed from fields and their values. As values, a Blob or File Object can be included in a FormData. Besides, ArrayBuffer is to represent a prevailing, and fixed length raw binary data buffer read from a file. The file variable, which can be either a Blob, File or FormData, is sufficient to determine an incoming request to be a file uploading request or not.

Therefore, CloudDLP overrides and hooks the XMLHttpRequest API by the injected JavaScript snippets for checking the argument type in the send method to identify file uploading requests.

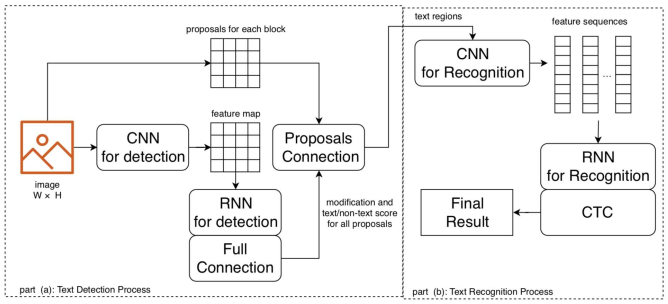


Fig. 2. The Model of Automatic Image Sanitization.

E. Automatic Image Sanitization

The first step of detecting privacy information in images is to extract texts in the form of pixels presents. This step presents a great challenge according to the variety of images coming from different cloud applications. To address this challenge, we adopts the automated method, named scene text reading. This model consists of two processes: text detection and text recognition. The text detection process is to locate the regions possibly containing text from the given images. The text recognition process focuses on transforming the text in detected regions into editable characters. The scene text reading model in this paper is based on deep learning, and its structure is shown in Fig. 5:

Text detection process: Different text detection models are designed for various purposes, attempting to solve unique difficulties in detecting scene text. For example, R2CNN [25], SegLink [26] and PixelLink [27] are designed to detect extremely long texts which have much larger height-width ratio. Aiming at the multi-oriented text, models such as [28], [29] and [30] have been proposed. Because texts in images in the cloud privacy preserving are usually horizontal and low-pixel, we choose the Connectionist Text Proposal Network (CTPN) model in [31], [32] and [33], which performs well on such images (like the dataset ICDAR2013) and has been wildly applied. The structure of CTPN is as follow: First, images are divided into small blocks with 16×16 pixels, and we define several proposals in each block. Then, we extract a feature map from these images with a convolutional neural network (CNN), in this paper, the CNN layer is built by referenced to the standard practice of VGG-16 and an additional convolution layer is added after the fifth convolution layer of CNN, in order to change the feature maps into a sequence. Then, a Recurrent Neural Network (RNN) is connected after the CNN to analyze a feature sequence. RNN is designed for processing sequence data, and it combines all the features in the line of this proposal. A full-connect network with 60 hidden units based on RNN predicts the location (center point coordinate), size (the fixed-width of proposal) and score (whether this proposal is text or non-text) for each proposal defined. The RNN we adopted in this paper is a bidirectional Long Short Term Memory (LSTM) with 256 hidden units. With the prediction of RNN and the full-connect layer, we filter out proposals with a low score and connect the high score proposals into a text region.

Text recognition process: Text recognition tasks can also be classified into horizontal text recognition [34], [35] and curved text recognition [36]. At the post-processing of the text recognition, the output text may be revised by an established word list called 'lexicon.' Using lexicon can improve the accuracy of the recognition model to certain extent, but this method is not suitable for cloud privacy preserving task that do not have a predetermined word list. The text recognition model we chose in this paper called Convolutional Recurrent Neural Network (CRNN) [37], which is dedicated to analyzing horizontal text issues and achieve good results without lexicon. The structure of CRNN is as follows: Before inputting into the recognition process, all of the text regions are scaled to a height of 32 pixels by bilinear sampling. At the beginning of this process, we use a special CNN to extract the feature of the text region and directly transform these features into sequential vector representations. This CNN adopts 1×2 sized maxpooling windows instead of the 2×2 ones in the CNN layer, in order to directly get feature sequences from the text regions. After the CNN layer, a bidirectional LSTM is used in this layer as the replacement of traditional RNN to predict a label for each vector and output a label sequence. This label sequence may contain blanks and redundancies like '-c-ll-ouu-d'. At the end, Connectionist Temporal Classification (CTC) [38] is adopted to change the label sequence into the final result 'cloud'.

The recognized text will be recognized by the NER model mentioned in the Section F, if it is a sensitive data category, its pixel corresponding to the sensitive information will be blacking-out.

F. Automatic Textual Document Sanitization

Confidential data of companies and organizations, such as intellectual property, financial information, personal credit card data and other information depending upon the business, is often in the form of textual documents. The most intuitive way is to upload the documents to the cloud after encryption. Unfortunately, encrypted files can cause features on the cloud to be unusable. For example, online document editing and preview cannot be used after file encryption. Hence, document sanitization not only removes privacy-sensitive information

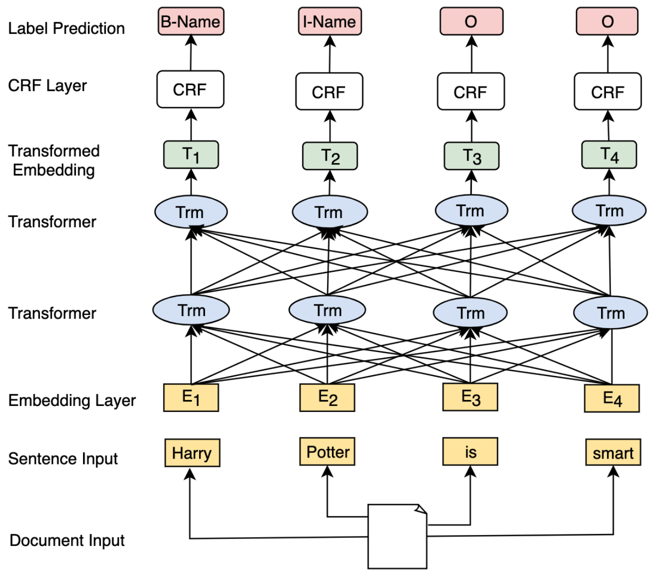


Fig. 3. The Model of Automatic Textual Document Sanitization

but also preserves document semantic information. Thus how to automatically localizing and sanitizing private content in textual documents is a serious challenge. Named Entity Recognition [15] is an information extraction technology to locate and classify elements in text into pre-defined categories such as the person names, organization, places, and so on. NER is applied in intelligently detecting sensitive data in documents.

Recently, BERT (Bidirectional Encoder Representations from Transformers) [39] is published by researchers at Google AI Language. It is trained on large text corpora with an unsupervised loss and has the state-of-the-art performance on NER. Thus we develop an automatic sanitization of textual documents using transfer learning on BERT. The network architecture is shown in Fig. 3. For a given document, we first break down the document into sentences using 'n', '.', '?', '!', and then split the sentence into tokens according to WordPiece tokenization. For a given token, its input representation is constructed by summing the corresponding token, segment and position embeddings which are implemented by Transformer and BERT. Furthermore, the first token of every sequence is always the special classification embedding ([CLS]) and end with another token ([SEP]). The BERTs model architecture is a multi-layer bidirectional Transformer encoder. We used the BERT base model, which has 12 Transformer blocks, a hidden size of 768, and the number of self-attention head as 12. The transformer can extract features efficiently, and it can be computed parallelly. The bidirectional Transformer can extract bidirectional representations by jointly conditioning on both left and right context in all layers. The outputs of the Transformer are the transformed embedding of every token. The outputs are fed into a linear-chain conditional random filed (CRF) [40]. The CRF is able to find the labeling sequence in a sentence with the highest probability. The determination of the token label can make use of both previous and subsequent

Text Detection Network

Layer Type	Configurations
Box FC & Score FC	hidden units:4×10 + 2×10
BiLSTM	hidden units:128
Additional Convolution	c: 512, k:3 × 3, s:1, p:1
Convolution 5	c: 512, k:3 × 3, s:1, p:1
MaxPooling 4	k:2 × 2, s:2
Convolution 4	c: 512, k:3 × 3, s:1, p:1
MaxPooling 3	k:2 × 2, s:2
Convolution 3	c: 256, k:3 × 3, s:1, p:1
MaxPooling 2	k:2 × 2, s:2
Convolution 2	c: 128, k:3 × 3, s:1, p:1
MaxPooling 1	k:2 × 2, s:2
Convolution 1	c: 64, k:3 × 3, s:1, p:1
Input	Images of any size (RGB)

Text Recognition Network

Layer Type	Configurations
CTC Transcription	—
BiLSTM 2	hidden units:256
BiLSTM 1	hidden units:256
Convolution 7	c: 512, k:2 × 2, s:1, p:0
MaxPooling 4	k:1 × 2, s:2
Convolution 6 with BatchNormalization	c: 512, k:3 × 3, s:1, p:1
Convolution 5 with BatchNormalization	c: 512, k:3 × 3, s:1, p:1
MaxPooling 3	k:1 × 2, s:2
Convolution 4	c: 256, k:3 × 3, s:1, p:1
Convolution 3	c: 256, k:3 × 3, s:1, p:1
MaxPooling 2	k:2 × 2, s:2
Convolution 2	c: 128, k:3 × 3, s:1, p:1
MaxPooling 1	k:2 × 2, s:2
Convolution 1	c: 64, k:3 × 3, s:1, p:1
Input	100×32 images, gray-scale

Fig. 4. Text Detection and Recognition Network

label information.

G. CloudDLP Implementation

So far, we have implemented CloudDLP on an enterprise gateway. Several companies are initially using our system to prevent data leakage. The implementation of CloudDLP proxy is in C++, based on a popular open-source proxy, Squid [41]. Squid is a caching proxy for the web supporting HTTP, HTTPS, and more. The JavaScript Snippets is easier to implement in JavaScript to override native JavaScript API. The Parser, the Sanitizer, and the Packer are implemented as microservices which are easier to scale. The deep learning model in CloudDLP is implemented in Python using Tensorflow [42] or Pytorch [43]. In the future, we will consider making the system open source.

IV. Experimental Evaluation

In this section, we first measure the performance of image and textual sanitization. We then discuss its effectiveness in a wide variety of popular applications. We evaluate both accuracy and time cost of CloudDLP system. Comparing with the image sanitization module and textual document sanitization module, the time cost of network communications can be ignored. So we only focus on the time cost evaluation of these two sanitization modules. Our experiments are carried out on a server for deep learning with an Intel(R) Xeon(R) Silver 4116 2.10GHz CPU, 64GB RAM, and an NVIDIA Quadro P4000 GPU.

A. Image Sanitization Performance

Deep learning based text reading model needs to be well trained with mass data to improving accuracy. In this paper, text detection process and text recognition process are trained respectively. The training data of text detection come from benchmark datasets CRTW-17 (ICDAR2017 Competition on Reading Chinese Text in the Wild) and ICPR Text Detection. The text recognition process is trained with synthetic data because of the lack of Chinese-English Mixed open source data. We choose suitable fonts, backgrounds and corpus to make synthetic data more real. Both the text detection model and the text recognition model use Stochastic Gradient Descent

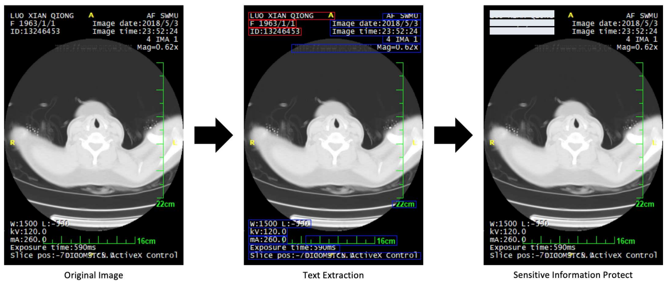


Fig. 5. The Result of Image Sanitization

(SGD) for CNN training and Back-Propagation Through Time (BPTT) for RNN training. We use ADADELTA [44] to control the learning rate and batch normalization technique to speed up the training process. The details of network parameters is shown in Fig. 4, where s is step size, p is the padding size, c is the number of channels, and k is the kernel size of convolution.

In these experiments, we adopt DetEval algorithm [45] to evaluate the performance of text detection, and Edit Distance to evaluate the accuracy of text recognition. The result is shown in Table. II. These models are tested on three mutually-exclusive data sets, images in these data sets are collected from three different wildly used cloud applications. The final result shows that the accuracy is about 93.4%, which is acceptable for information extraction. The average time-consuming is also shown in Table. II. The result shows that most of the time is in text detection. Fig. 5 shows image sanitization module detects and sanitize sensitive information such as person name, age, and ID number in medical image pictures.

B. Textual Document Sanitization Performance

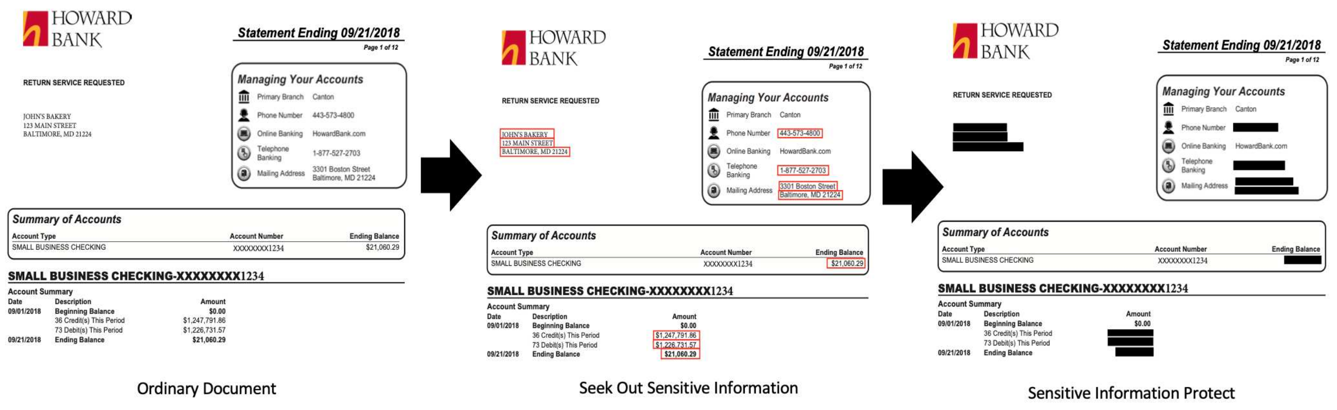
The main data set used to evaluate the model of textual document sanitization is the 2014 i2b2 de-identification challenge data set [46]. CloudDLP need to strip the medical records of any protected health information (PHI). The i2b2-PHI categories are shown in Table. I. The data set consists of 1,304 patient progress notes for 296 diabetic patients and contains 56,348 sentences with 984,723 separate tokens (of which 41,355 are separate PHI tokens), which represent 28,867 separate PHI instances. We used the standard metrics such as precision, recall, and F1 as defined in Equation 1-3, to measure the performance of the model. Moreover, we also compute these three metrics for each PHI category.

$$Precision = \frac{No.\ of\ correctly\ identified\ PHI\ Tokens}{No.\ of\ identified\ as\ PHI\ Tokens} \tag{1}$$

$$Recall = \frac{No.\ of\ correctly\ identified\ PHI\ Tokens}{TotalNo.\ of\ PHI\ Tokens} \tag{2}$$

$$F_1 = 2 * \frac{precision\ *\ recall}{precision\ +\ recall} \tag{3}$$

We used a BERT-based model which have 12 layers of Transformer Blocks, 768 hidden size, 12 self-attention heads, and 110M parameters. We evaluated the model on i2b2-PHI


HOWARD BANK

Statement Ending 09/21/2018

Page 1 of 12

RETURN SERVICE REQUESTED

JOHN'S BAKERY
123 MAIN STREET
BALTIMORE, MD 21224

Managing Your Accounts

Primary Branch	Canton
Phone Number	443-573-4800
Online Banking	HowardBank.com
Telephone Banking	1-877-527-2703
Mailing Address	3301 Boston Street Baltimore, MD 21224

Summary of Accounts

Account Type	Account Number	Ending Balance
SMALL BUSINESS CHECKING	XXXXXXXX1234	$21,060.29

SMALL BUSINESS CHECKING-XXXXXXXX1234

Account Summary

Date	Description	Amount
09/01/2018	Beginning Balance	$0.00
	36 Credit(s) This Period	$1,247,791.86
	73 Debit(s) This Period	$1,226,731.57
09/21/2018	Ending Balance	$21,060.29

Fig. 6. The Result of Textual Document Sanitization

TABLE I
PHI CATEGORIES ON I2B2

HIPPA	i2b2
Name	Patient, Doctor, Username
Profession	Profession
Location	Street, City, State, Country,Zip,Hospital,Organization
Age	Age
Date	Date
Contact	Phone,Fax,Email,URL,IP Address
ID	Medical Record,ID No,SSN,License No

TABLE II
IMAGE SANITIZATION PERFORMANCE

Dataset	Text Detection				Text Recognition	
	Precision	Recall	F-Score	Time (s)	Accuracy	Time (s)
Image Set 1	76	73	75	4.686	95	0.415
Image Set 2	79	81	80		92	
Image Set 3	77	76	76		93	

TABLE III
IMAGE SANITIZATION PERFORMANCE

Category	Precision	Recall	F1-Score
Username	0.9468	0.9674	0.9570
Doctor	0.9516	0.9449	0.9483
Patient	0.9402	0.9391	0.9396
Profession	0.7727	0.8293	0.8000
State	0.8186	0.8930	0.8542
Street	0.8904	0.9559	0.9220
Age	0.9632	0.9819	0.9725
Date	0.9830	0.9750	0.9750
Phone	0.8900	0.9490	0.9185
Zip	0.9568	0.9568	0.9568
ID	0.8093	0.8220	0.8156
MedicalRecord	0.9731	0.9827	0.9779
average	0.9796	0.9836	0.9816

categories for the i2b2 data set based on token-level labels. Table. III, summarizes the performance of our model on i2b2-PHI categories. As we can see in Table. III, our model achieves an impressive precision of 0.9796 and a recall of 0.9836. Fig. 6 shows textual document sanitization module detects and sanitize sensitive information such as person name, phone number, mailing address, and so on in the pdf file.

C. Case Study

We test CloudDLP with typical cloud applications and observe the potentially affected functionalities including search, online document editing, document previews, and thumbnail previews. The result shows that CloudDLP retains prominent functionality of these applications while protecting sensitive critical data. We tested CloudDLP with typical browser-based cloud storage applications such as **Gmail**, **Dropbox**, **Box**, **OneDrive**, **Google Drive**, and **Mega.nz**. CloudDLP successfully detects and sanitizes sensitive information in files uploaded to the cloud. Online editing, previewing, and sharing of documents can still be used normally because the sanitization of a small amount of sensitive information does not destroy the document structure and makes it impossible to parse in the cloud. On using CloudDLP with office applications, namely **Salesforce**, **Google Docs**, and **Slack**, the uploaded files of Salesforce and Google Docs can be protected by CloudDLP. Importing a file into a trading repository and report forms may be wrong with Salesforce. The sanitization data, if it is a numeric type, would affect the results of statistical analysis in the report. Fortunately, users can choose whether to enable the protected mode by clicking the hovering button in the head of the web page.

V. CONCLUSION

In this paper, we present CloudDLP, a generic approach that transparently and automatically performs data sanitization on various cloud applications while preserving most functionalities of the cloud. As a result, CloudDLP can help enterprises effectively prevent core data leakage, as well as can ensure sensitive data security and regulatory compliance in the cloud. In addition, our experimental results show that CloudDLP has fairly low overheads and can support practical use in many real-world applications.

VI. Acknowledgement

This research is supported by the National Key R&D Program of China (Grant No.2016YFB0800803) and the National Natural Science Foundation of China (Grant No.61872110).

References

[1] "Theres a hole in 1,951 amazon s3 buckets." [Online]. Available: https://community.rapid7.com/community/infosec/blog/2013/03/27/1951-open-s3-buckets

[2] "Top defense contractor left sensitive pentagon files on amazon server with no password." [Online]. Available: http://gizmodo.com/top-defense-contractor-left-sensitive-pentagon-files-on-1795669632

[3] "Patient home monitoring service leaks private medical data." [Online]. Available: https://mackeepersecurity.com/post/patient-home-monitoring-service-leaks-private-medical-data-online

[4] "198 million americans hit by largest ever voter records leak." [Online]. Available: http://www.zdnet.com/article/security-lapse-exposes-198-million-united-states-voter-records/

[5] "Spirion." [Online]. Available: https://www.spirion.com/

[6] "Cuspider." [Online]. Available: https://www.spirion.com/

[7] "Ciphercloud: cloud services adoption while ensuring security, compliance and control." [Online]. Available: https://www.ciphercloud.com/

[8] W. He, D. Akhawe, S. Jain, E. Shi, and D. Song, "Shadowcrypt: Encrypted web applications for everyone," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. New York, NY, USA: ACM, 2014, pp. 1028–1039. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660326

[9] B. Lau, S. Chung, C. Song, Y. Jang, W. Lee, and A. Boldyreva, "Mimesis aegis: A mimicry privacy shield–a systems approach to data privacy on public cloud," in *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, 2014, pp. 33–48.

[10] S. Ruoti, J. Andersen, T. Monson, D. Zappala, and K. E. Seamons, "Messageguard: A browser-based platform for usable, content-based encryption research," *CoRR*, vol. abs/1510.08943, 2015. [Online]. Available: http://arxiv.org/abs/1510.08943

[11] "Cloud access security brokers." [Online]. Available: https://www.gartner.com/it-glossary/cloud-access-security-brokers-casbs/

[12] "Skyhigh: 9 cloud computing security risks every company faces." [Online]. Available: https://www.skyhighnetworks.com/cloud-security-blog/9-cloud-computing-security-risks-every-company-faces/

[13] K. Kaur, I. Gupta, and A. K. Singh, "A comparative study of the approach provided for preventing the data leakage," *Int. Journal of Network Security & Its Applications (IJNSA)*, vol. 9, no. 5, pp. 21–33, 2017.

[14] E. Costante, D. Fauri, S. Etalle, J. Den Hartog, and N. Zannone, "A hybrid framework for data loss prevention and detection," in *2016 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2016, pp. 324–333.

[15] J. M. Gomez-Hidalgo, J. M. Martin-Abreu, J. Nieves, I. Santos, F. Brezo, and P. G. Bringas, "Data leak prevention through named entity recognition," in *2010 IEEE Second International Conference on Social Computing*. IEEE, 2010, pp. 1129–1134.

[16] "Insights you want.inspiration you need." [Online]. Available: http://www.google.com/insights/search/

[17] Y. J. Ong, M. Qiao, R. Routray, and R. Raphael, "Context-aware data loss prevention for cloud storage services," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*. IEEE, 2017, pp. 399–406.

[18] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, "Cryptdb: protecting confidentiality with encrypted query processing," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. ACM, 2011, pp. 85–100.

[19] R. A. Popa, E. Stark, S. Valdez, J. Helfer, N. Zeldovich, and H. Balakrishnan, "Building web applications on top of encrypted data using mylar," in *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, 2014, pp. 157–172.

[20] "Virtru: Email encryption and data security for business privacy." [Online]. Available: https://www.virtru.com

[21] E. Wittern, A. T. Ying, Y. Zheng, J. Dolby, and J. A. Laredo, "Statically checking web api requests in javascript," in *Software Engineering (ICSE), 2017 IEEE/ACM 39th International Conference on*. IEEE, 2017, pp. 244–254.

[22] https://developer.mozilla.org/en-US/docs/Web/API/Blob.

[23] https://developer.mozilla.org/en-US/docs/Web/API/FormData.

[24] https://developer.mozilla.org/en-US/docs/Web/API/ArrayBufferView.

[25] Y. Jiang, X. Zhu, X. Wang, S. Yang, W. Li, H. Wang, P. Fu, and Z. Luo, "R2cnn: Rotational region cnn for orientation robust scene text detection," *arXiv preprint arXiv:1706.09579*, 2017.

[26] B. Shi, X. Bai, and S. Belongie, "Detecting oriented text in natural images by linking segments," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2550–2558.

[27] D. Deng, H. Liu, X. Li, and D. Cai, "Pixellink: Detecting scene text via instance segmentation," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[28] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai, "Multi-oriented text detection with fully convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4159–4167.

[29] C. Yao, X. Bai, and W. Liu, "A unified framework for multioriented text detection and recognition," *IEEE Transactions on Image Processing*, vol. 23, no. 11, pp. 4737–4749, 2014.

[30] Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao, "Oriented response networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 519–528.

[31] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao, "Detecting text in natural image with connectionist text proposal network," in *European conference on computer vision*. Springer, 2016, pp. 56–72.

[32] D. Wu, R. Wang, P. Dai, Y. Zhang, and X. Cao, "Deep strip-based network with cascade learning for scene text localization," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 1. IEEE, 2017, pp. 826–831.

[33] X. Zhu, Y. Jiang, S. Yang, X. Wang, W. Li, P. Fu, H. Wang, and Z. Luo, "Deep residual text detection network for scene text," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 1. IEEE, 2017, pp. 807–812.

[34] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep structured output learning for unconstrained text recognition," *arXiv preprint arXiv:1412.5903*, 2014.

[35] ——, "Reading textjin'tia in the wild with convolutional neural networks," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 1–20, 2016.

[36] C. Luo, L. Jin, and Z. Sun, "A multi-object rectified attention network for scene text recognition," *arXiv preprint arXiv:1901.03003*, 2019.

[37] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 11, pp. 2298–2304, 2017.

[38] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376.

[39] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[40] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[41] "Squid." [Online]. Available: http://www.squid-cache.org/

[42] "tensorflow." [Online]. Available: https://www.tensorflow.org/

[43] "pytorch." [Online]. Available: https://pytorch.org/

[44] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[45] C. Wolf and J.-M. Jolion, "Object count/area graphs for the evaluation of object detection and segmentation algorithms," *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 8, no. 4, pp. 280–296, 2006.

[46] "i2b2 dataset." [Online]. Available: https://www.i2b2.org/NLP/DataSets/