A policy-based containerized filter for secure information sharing in organizational environments

J.L.Gonzalez-Compean, Oscar Telles, Ivan Lopez-Arevalo,
Miguel Morales-Sandoval, Victor J. Sosa-Sosa, Jesus Carretero

PII: S0167-739X(18)31177-4
DOI: https://doi.org/10.1016/j.future.2019.01.002
Reference: FUTURE 4695

To appear in: *Future Generation Computer Systems*

Received date : 16 May 2018
Revised date : 18 November 2018
Accepted date : 1 January 2019

Please cite this article as: J.L.Gonzalez-Compean, O. Telles, I. Lopez-Arevalo et al., A policy-based containerized filter for secure information sharing in organizational environments, *Future Generation Computer Systems* (2019), https://doi.org/10.1016/j.future.2019.01.002

A policy-based containerized filter for secure information sharing in organizational environments

J.L.Gonzalez-Compean, Oscar Telles[1], Ivan Lopez-Arevalo, Miguel Morales-Sandoval

Cinvestav-Tamaulipas, Ciudad Victoria, México

Victor J. Sosa-Sosa, Jesus Carretero

Carlos III Madrid University, ARCOS research group, Spain

Abstract

In organizational environments, sensitive information is unintentionally exposed and sent to the cloud without encryption by insiders that even were previously informed about cloud risks. To mitigate the effects of this information privacy paradox, we propose the design, development and implementation of SecFilter, a security filter that enables organizations to implement security policies for information sharing. SecFilter automatically performs the following tasks: a) intercepts files before sending them to the cloud; b) searches for sensitive criteria in the context and content of the intercepted files by using mining techniques; c) calculates the risk level for each identified criterion; d) assigns a security level to each file based on the detected risk in its content and context; and e) encrypts each file by using a multi-level security engine, based on digital envelopes from symmetric encryption, attribute-based encryption and digital signatures to guarantee the security services of confidentiality, integrity and authentication on each file at the same time that access control mechanisms are enforced before sending the secured file versions to cloud storage. A prototype of SecFilter was implemented for a real-world file sharing application that has been deployed on a private cloud. Fine-tuning of SecFilter components is described and a case study has been conducted based on document sharing of a well-known repository (MedLine corpus). The experimental evaluation revealed the feasibility and efficiency of applying a security filter to share information in organizational environments.

Keywords: Cloud security, Risk assessment, Mining, Multi-level security, Virtual Containers
2010 MSC: 00-01, 99-00

1. Introduction

Sharing information tools are keys for organizations to exchange contents with users, consumers and partners in an anytime and anywhere manner [1]. Cloud storage has become a solution for organizations to implement sharing information systems in cost-efficient manner, mainly because of the outsourcing and pay-as-you-go models associated to cloud technology. Nevertheless, security aspects such as integrity of data, privacy of contents, confidentiality and secure access control to data, still represent an obstacle for organizations to adopt cloud technology in a confident manner [2, 3, 4]. Moreover, government regulations and standards have been established on few past years for organizations to manage and preserve sensible information [5, 6] in a secured manner. In order to take advantage of the outsourcing economic model of the cloud and to observe the government and organizational regulations for the protection of information, organizations have increased the investment to implement strategies to protect their information assets [7, 8, 9], and to implement security solutions, both inside and outside of the organization ambit (e.g. the cloud). Encryption has become an attractive solution for organization to ensure privacy over the content shared by users through the cloud [10, 11, 8, 9]. Some of those schemes are based

☆Fully documented templates are available in the elsarticle package on CTAN.

on attributes and oriented to implement role-based access strategies to enable fine-grained sharing of users' files in the cloud [12, 13, 14]. However, studies have shown that half of information is sent by the users to the cloud either without encryption or encrypted with a security level that is not adequate for organization policies [10, 11].

Despite training programs conducted for users to avoid risky behaviors when managing information inside and outside of the organizations, and even when users in the organizations are informed about security risks associated to cloud technology, empirical research studies have identified that security violation incidents are produced by risky behavior of the users [15, 16, 10, 11]. This scenario results in a security challenge when the size of organizations grows and there are more users sharing files and involved in sharing information workflows [7]. Moreover, studies reveal that organizations should face up this challenge, not only by implementing information security solutions focused on confidentiality and privacy issues, but also by establishing and implementing security controls based on security policies [17].

This paper presents the design, development and implementation of *SecFilter*, a policy-based security filter deployed on virtual containers that enables organizations to establish security policies and controls over information sharing operations. SecFilter acts as an intermediary between either synchronized folders or file sharing applications and cloud storage services. It implements a security policy that considers four phases developed in four modules encapsulated into virtual containers. I) In the *Interception of File Phase*, the administrators define a storage path where users deposit files going to be stored in a cloud storage service (e.g. a synchronized folder or organizational shared directory). When SecFilter is deployed in the user computers, a module automatically intercepts the files sent to the folder monitored by SecFilter. II) In the *Risk Assessment Phase*, the organization define sensitive criteria and assigns a weight/rank to each defined criterion, which in the current module includes context and content criteria. In a regular operation of SecFilter, this module identifies whether sensitive criteria are found or not within the intercepted files by using a verification system based on mining techniques [18, 19]. This module calculates a risk scoring for each file in a sharing operation, depending on the criteria found, and this score is mapped to a risk level (e.g. high, medium, and low)[1]. III) In the *Mitigation Phase*, a security module performs the encryption of the file depending on the risk scoring detected in the previous phase by using a multi-level security engine, which uses symmetric encryption to ensure privacy and confidentiality [20], attribute-based encryption to ensure confidentiality and access control [21, 22, 13], and digital signatures to ensure integrity and authentication [23, 13]. Security level can be changed on-the-fly and the security level for the three types of encryption is determined based on the risk level detected on the file content/context (The bigger the risk level, the larger the size of the encryption keys) and the access control attributes are chosen for each file depending on the context (The attributes of the users included in a sharing operations are on-the-fly defined and used in the encryption/decryption of each file). IV) In the *Storage Phase*, a storage module sends the digital envelopes of analyzed files to, either a shared folder synchronized with a cloud storage service, or directly to the storage service (depending on the apps included in the storage module).

A prototype was developed by using the modular design of security filter, deployed on a virtual containers scheme, and implemented in a real-world file sharing application. The prototype was evaluated through case studies resembling users sharing documents from MedLine corpus with other users through a private cloud. A set of digital products of a satellite imagery was also processed by SecFilter to show the flexibility of managing different file formats. The evaluation revealed the feasibility and efficiency of applying a policy-based filter to sharing information environments.

SecFilter modules (one per phase of the security policy) can be configured by the IT administrators of the organizations through a GUI, which also enables organizations to identify the contents found in the file secured by SecFilter.

In summary, the main contribution of this paper is:

A security information method combining smart and reactive analytic solutions with encryption and security policy schemes for organizations to face up the challenge of security information paradox instead only using an encryption solution. This method is materialized in an

[1]The filter can be configured to consider either one or n criteria in the risk assessment phase.

implementation called SecFilter, a containerized modular security filter implementing security policies based on context/content identification of sensible criteria. The modularity of SecFilter design enables developers to build parallel patterns in the virtual container of SecFilter, improving the performance of analysis and ensuring methods and making feasible the deployment of SecFilter in real-world information sharing scenarios. The discovery of sensible criteria in sharing patterns is feasible as a result of the implementation of this solution, which provides organization with a big picture of the sharing information environment.

The outline of this paper is as follows. Section 2 presents existing works related to our solution. Section 3 presents the design principles, architecture, and major components of the security filter. Section 4 shows the prototype implemented following previous design principles and architecture. Section 5 describes the experiments run to test the system, metrics used to evaluate our proposal, and discusses the results obtained from the experiments and compares them against other approaches. Finally, Section 6 presents conclusions of our work and some future research lines.

2. Related Work

Cloud computing and Cloud data storage were seen as a promising business paradigm for information sharing. An example was shown by Rosenthal [24] for biomedical information sharing. However, data security has always been a major concern in Cloud, as shown in several studies [25, 26] that concluded that Cloud providers cannot be treated as a trusted third parties because of its semi-trust nature. Due to this reason, the traditional security models cannot be straightforwardly generalized into cloud based information sharing frameworks. As a result, there has been several proposals to achieve secure data sharing in the cloud, most of them based on data encryption at different points and with different features. As an example, Li et al proposed in [27] a scalable and secure sharing of personal health records in Cloud computing using attribute-based encryption. Other works have proposed to extend those solutions to key-aggregate cryptosystems [28] or multi-owner data sharing for dynamic groups in the Cloud [29].

Symmetric cipher is the Advanced Encryption Standard (AES) [20] and ABE is a Ciphertext Policy Attribute Based Encryption (CP-ABE) [30] used as (non conventional) public key cryptography have been a popular solution for users to secure data with confidentiality protection. Nevertheless, the concerns expressed by cloud storage users about security seem to be better addressed by using attribute based encryption [31, 32, 33, 34], specially to enable secure sharing. In these works, the concept of digital envelop is used as a container for secure information transmission, thus providing reliable an efficient digital envelopes is critical to those solutions.

A realization of digital envelopes from ABE based on symmetric cryptography (AES) to encrypt data and policy-based attribute to encrypt the AES key by using public key cryptography has been [22] proposed. In this model, only the recipient of the digital envelope, with its private key, can decrypt the key with which to decrypt the data. Different from public key cryptography, the encryption in ABE scheme is performed by using a policy instead a key. The policy is constructed as a Boolean function of different attributes. Users are assigned with a specific set of attributes, and their decryption capabilities will depend on if their attributes satisfy or not the encryption policies.

Although theoretical security schemes based on risk assessment have been proposed for encrypting files in automatic manner [35], in practice, only solutions using attribute-based encryption are available to establish access controls over the information before sending files to the cloud [13, 36, 37, 14, 38]. Nevertheless, these works were designed mainly as a static solution by applying either a given security level to all the files or a given security key size to all the sharing operations. In turn, SecFilter enables organizations to integrate different types of criteria in the risk assessment, to secure data with different types of cryptographic key sizes that are automatically chosen by the filter depending on the risk scoring levels.

The security of smart grid networks and sensor networks also have received attention form the industry and the Fully Homomorphic Encryption schemes [39, 40, 41] used to face up these issues result in promissory solution to be applied to the other similar problems as networks of organization sharing information.

Nevertheless, the organizations cannot face up the information security issues by using only encryption solutions as the human factor has also been observed as a key, and challenging, component for the safety of the organization's data [17]. It was detected, in a consistent manner from 2014 to 2017, that major incidents were produced because of "private or sensitive information unintentionally exposed" [42, 8, 9]. This behavior, called information privacy paradox in the literature [43], is becoming critical in organizational dynamics [16, 15, 5] because organizations relegate the responsibility of ensuring files to the users. This situation represents a potential critical issue for organizations, specially in scenarios of information sharing, which are quite common in the interactions of the organizations with consumers, partners and workers/users [44]. Organizations face up the challenges of verifying that users establish controls on the sensitivity of the content to be shared (e.g. sensitive keywords and/or topics for company/organization), the access control to the shared content (i.e. granting access only to a valid group of users), and the ambit where users are sharing the content (e.g. the branch/level in the organization chart where users are allowed to perform sharing operations).

In this context, combining smart and reactive solutions with encryption schemes seem to be more suitable for the organizations to face up this type of challenge than only using an encryption solution. SecFilter automatically makes decisions and all the intercepted files are secured at least with the default security level (by using a key of 128 bits size), which reduces omissions in the encryption of files by the users.

Regards to the application of data analysis on information to be stored on cloud computing systems (without security schemes), most of the approaches are about storing images [45], biomedical information [46], and clinical records [47]. According to our knowledge, there are not similar works to our proposal for storing data based on analysis of content/context and a security scheme. Some of the most related works are the described next. Simske and Balinsky [48] proposed a system for document ensuring based on policies on tags from documents and document-handling user actions. For this, the system uses some predefined tags from document metadata and internal parts of documents. Each document is logically divided according to policies to be applied based on tags found in documents; with this the security level of each part is determined. The system can be used into an organization by using local storage or cloud-based storage. Calero [49] proposed an authorization system for distributed environments. This system exposes the semantics of information to be protected. The proposed representation consists of an information model based on a logic formalism and a formal language for describing the semantics of the information. It was implemented on a prototype of system based on authorization methods through REST interfaces. Shatnawi et al. [50] described an approach for maintaining integrity and non-repudiation of collaborative Microsoft Word documents (XML documents). This mechanism maintains logs of modifications, forensics information, and users' digital certificates of offline documents, which can be later shared (email, personal storage devices, local storage, cloud storage). Wua et al. [51] described a distributed framework for text processing for building discriminator services. In this framework injection, indexing, storage, and processing of text is executed on the server-side platform, while the information is consumed remotely by several mobile-side clients. The framework takes passages from web pages to produce valuable information to consumers. The proposal was evaluated on an hypothetical case study by using Hadoop.

The main problem with most of the solutions reported in the literature is that they are adapted to a specific problem and type of data (e.g biological, business, etc.), which somehow limits the validity of the existing solutions. To cope with this problem, there has been proposals recently to provide more general frameworks. For example, Xue et al. [52] proposed a secure group sharing framework for public cloud that avoids to have sensitive data being exposed to attackers and the Cloud provider by working on the client side combining proxy signature, enhanced TGDH and proxy re-encryption together into a protocol. Another example oriented towards urban data sharing in smart cities is shown in [53], but both without analyzing the content of the files.

Still in these last works, a more generic solution does not consider the content of data to determine, in an automatic way, which security requirements and controls must be applied to that content before its outsourcing to the cloud. The frameworks proposed in this paper, SecFilter, has new features compared to the previous solutions as it is client-based, modular, and content- and context-based, as we show in the next sections.

3. Design principles and architecture of SecFilter

In this section, we describe the design principles used to implement the four phases of the security policy and the development details of the components developed for the security filter. We also describe the adaptation of SecFilter to the operations of an information sharing application used in organizational environments. We also described the development details of SecFilter components such as criteria verification, risk assessment and security/mitigation. Interception and delivery modules are described when the prototype is described.

SecFilter has been designed as a modular pipeline system where each module of the pipeline implements a phase of the security policy. With this model, an organization can add/remove modules to/from the security filter depending on specific requirements.

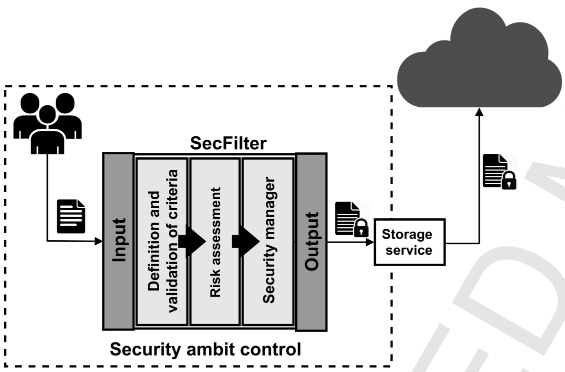


Figure 1: Conceptual representation of SecFilter architecture.

Figure 1 shows the conceptual representation of SecFilter. As it can be seen, the pipeline of modules developed in this filter implements the phases considered in the security policy defined in this paper. The *data interception module* (see input in Figure 1) is in charge of monitoring either the files arriving to the filter (in *pseudo-shared folders*) or I/O operations produced by a sharing application.

The intercepted files are sent to the next module, which verifies whether a file includes sensitive information in its context and/or content. This is performed by using two aspects: criterion sensitivity and user privileges. It is assumed that organizations define a list of sensitive criteria (e.g. topics, sets of characteristics, groups of data, patterns, etc.) - *a list of clusters*- and a list of users hierarchically categorized. A weight for each aspect is also defined (e.g. 60% for sensitivity of content criteria and 40% for context criteria). Two submodules enables the filter to verify the criteria established by the organizations. The first one is a mining tool that performs content analysis by preprocessing and processing tasks to create clusters that allows the filter to determine if a sensitive criterion is accomplished in the content of each analyzed file. The second one is a parsing tool created to determine the context criteria (e.g. the place of a user in an organization chart). In both cases key-value maps are created. In the case of content, a map including the id-file and a sensitive criterion is created as well as a map including contextual information (e.g. an ID-User and the level/branch of that user(s) in an organization chart). It is important to note that this module delivers as many maps as criteria defined by the organization. For instance, an organization could choose that Secfilter only considers for the files their context, their content or both. More criteria could be considered in this phase by adding more ad-hoc modules.

The *risk assessment module* receives the maps previously computed and use them to calculate a risk scoring level together with the list of sensitive criteria defined by the organization (that includes the weight of each criterion assigned by the organization). A comparison of the received maps with the lists of sensitive aspects is performed to calculate a risk level (in this paper three levels are considered: *high, medium and low*). This module creates a key-value map including control information such as the ID of the analyzed file (ID-File) and an ID of the action (ID-Action) required to mitigate the identified risk level.

The *security module* receives the action maps produced in the risk assessment module, identifies the security level to be assigned to protect files and applies the security mechanisms over the data. A multi-level security engine selects on-the-fly the size of the key depending on the action maps and proceeds to encrypt the files associated to the ID-File. In this paper, keys of 128, 192 and 256 bits are considered by the engine to manage low, medium and high risks respectively. Encrypted data are delivered as a secure digital envelope: the data are encrypted with symmetric cryptography and attribute-based encryption protects the data encryption keys by using policies in terms of the of file's context (e.g. attributes of a group of users commonly defined by the organizational dynamics such as branch, level and group considered in a sharing operation). For these policies, combinations of operands can be used for ensuring files in sharing operations (i.e. branch1 AND level1 and branch1 OR level1) so that only valid users could decrypt files.

The *file delivery/storage module* receives the secured version of the files, in the form of secure digital envelopes, and either sends them to the shared folder synchronized with a cloud storage service or directly to the Cloud.

Sharing operations are feasible over this model as users with valid attributes and credentials not only can retrieve files, but also decrypt them.

3.1. Development of module for the definition and validation of data security criteria

The development details of criteria verification SecFilter is described in this section in a phase of content analysis and another of context analysis.

Today, storing textual information became a need for several users, organizations, and companies. Nearly 95% of data stored on digital media are unstructured [54, 55]. From this, most of the available information is text (documents, web pages, emails, technical documents, scientific articles, clinical records, etc.). Also, more and more non-textual files (image, sound, video, spreadsheet, etc.) contain textual metadata or descriptions for cataloging purposes.

In this context, we included a text mining module as a tool to identify content criteria in the files at this stage of the SecFilter to show the feasibility of this solution. Nevertheless, this module can be replaced with other content analysis module available in the literature to identify sensible criteria (e.g. machine learning for images [56, 57, 58], video [45, 59] or sound [60, 61]).

3.1.1. Development of module for identifying sensible topics in file content

The criteria module is in charge of analyzing each document and determining the topic that statistically represents a document in each data sharing operation.

Figure 2 shows the conceptual representation of the definition and validation of criteria phase for the topic detection task. This task comprises the generation of the topics lexicon and classification. The fist one aims at identifying topics from the set of documents. This approach is based on keyword co-occurrence mapped on a graph representation. The

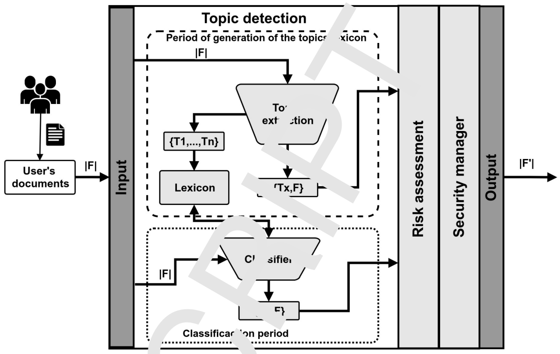


Figure 2: Definition and validation of criteria phase.

process is composed of four stages: pre-processing, words graph building, topic features extraction, and topics assignment to document. These stages are described below.

First, documents from the set are pre-processed by eliminating stopwords (words without semantic meaning: pronouns, articles, prepositions, etc.) and applying stemming (reduce a set of similar -syntactically- words to its common root).

In the second stage, remaining words from each document are extracted and joined to construct a *lexicon*. The words of the lexicon (terms, nouns and verbs) are used to build a graph of words. These words are identified into each document to determine its neighbor words for defining its co-occurrence. Each node in the graph corresponds to a word in the document and edges represent the co-occurrence of its corresponding connected words, a weight is assigned to each edge according to the count of co-occurrences. An edge is added to the graph when each pair of words co-occur in at least one document. For each word in a node, the document frequency is computed. The document frequency is also computed for each edge, which denotes the count of documents containing the co-occurrence of two connected words. Nodes and edges with low document frequency are eliminated from the graph.

In the third stage, *topic features* are extracted based on connected nodes. This approach is based on the Distributional Similarity approach on documents [62, 63]: it will tend to have the same or related meanings when two words co-occur on a sentence. The co-occurrence of two words denotes a topic relationship between them. A topic feature is a word related to a topic; as a result, a topic includes several words with related meaning. The count of co-occurrence of two words will also increase because of the increment in the number of documents related to that topic. The relationship of a word to a topic is obtained by clustering the words on the graph, which is carried out by using the Betweenness Centrality algorithm for community detection on graphs [64]. This algorithm determines the stronger edges between communities in a network. Betweenness Centrality for an edge is the count of the shortest paths for all pairs of nodes in the network that pass through such edge. This means inter-community edges will obtain a high score since the shortest paths between nodes from different communities will always pass through these edges.

In the fourth stage, topics are assigned to documents. Each community of nodes (cluster of words) in the graph represents a *topic*, topic features are represented into a feature vector. The likelihood of a topic being associated with a document is determined by the cosine similarity applied to the vector of the document and the feature vector. Thus, most common words (topic features) for a topic will have the highest weight. The more closely the word is representative of a topic, the more it will co-occur with other words that are also associated to such a topic. As a result, documents are assigned a likelihood over topics, taking into account that a document may have multiple topics. Algorithm 1 shows the process of the period of generation of the topics lexicon.

In practice, this module considers three configurations: The first one to build the lexicon of topics (initialization of the service), the second one for classifying files on-the-fly and the last one to update the lexicon by performing an another revision of set of files.

A lexicon of topics is created when SecFilter processes the first set of files. In this procedure, each file analyzed by SecFilter is mapped to a topic (indexed) and sent to the cloud. At this point, it is expected that few new topics being discovered, and the challenge arise for assigning a topic to new documents (non-existent documents into the original set of documents). In order to manage this challenge, the classification on-the-fly configuration of SecFilter was developed by applying a Bayesian approach to the classification process.

In the classification task, the Bayesian approach is based on *a priori* probabilities of topics and words into the graph generated in the topic identification stage. That is, probabilities are computed for: a) each existent topic $P(t_i)$ and b) the words of an existent topic $P(w_j|t_i)$. All the existent topics and all the words per topic in the graph are considered. This constitutes the training stage of the classifier. Each new document is pre-processed by eliminating stopwords and applying stemming for reducing the number of its words. The remaining words are used for assigning the topic to such document according to the following criterion (Eq. 1):

$$Topic_{NewDoc} = argmax_{t_i \in T}(P(t_i) \prod_{w_j \in D} P(w_j|t_i)) \quad (1)$$

Algorithm 1 Generation of the topics lexicon

Require: Documents d
for all document d_i **do**
  Delete stopwords from d_i
  Stemming (d_i)
  Extract words from d_i as keywords w_i
  Create one node for each keyword w_i
  Create one edge $e_{i,j}$ for each pair of co-ocurring keywords w_i and w_j
end for
keygraph ← buildGraph(nodes,edges)
for all node w_i **do**
  if ($frequency(w_i) < node\_min\_df$) **then**
    deleteNode (w_i)
  end if
end for
for all edge $e_{i,j}$ **do**
  if ($frequency(e_{i,j}) < edge\_min\_df$) **then**
    deleteEdge $(e_{i,j})$
  end if
end for
topics T ← Partition the keygraph into communities
for all topic $t \in$ T **do**
  Topic feature vector $ft_i \leftarrow Communiti_i(keywords)$
end for
for all topic t **do**
  similarity ← cosineSimilarity(topic t, document d)
  if (similarity > $min\_doc2topic\_similarity$) **then**
    $documentTopicMap.put(d, t)$
  end if
end for
return $documentTopicMap$, $topics$

Where T is the whole set of topics, D is a document, and w is a remaining word in document D.

$P(t_i)$ and $P(w_n|t_i)$ are retrieved from the training stage for each remaining word into the document D. In this equation, is interpreted that the assigned topic to a file is the one with the highest score based on the probability that a topic exists in the graph of words and the probabilities of words in that file appear in such a topic. In this way, the remaining words of a document contribute to define which topic can be assigned to such a document.

The method ensures that all new documents added to the set of documents will have a topic assigned. Algorithm 2 shows the process of the classification period.

Algorithm 2 Classification period

Require: documents d, topics $= \{t_1, t_2, \ldots, t_m\}$

```
class ← 0
% Training stage of the classifier
for all topic t_i do
    getProbability(t_i)
    for all word wt_i ∈ t_i do
        getProbability(wt_i)
    end for
end for
% Classification of a new document
Delete stopwords from d_i
keywords ← Stemming(d_i)
for all topics t_i do
    probability ← 1
    for all keyword w_i do
        probability = probability × probability(w_i)
    end for
    probability = probability × probability(t_i)
    if probability > probability(t_i) then
        class = probability
        IdentifiedTopic = t_i
    end if
end for
return IdentifiedTopic
```

Updating the lexicon by analyzing another set of files is possible, but it is expected to be performed in the long term. This situation is explained in evaluation section in more detail.

3.1.2. Development of module for identifying file context

In this section we describe a tools developed for identifying the context of a file through an organizational hierarchy. In this context, this tool receives the an organizational chart as input and an engine search for the document owner and the users to be considered in a given sharing operation depending on the organizational chart provided by the organization in configuration time. This tool creates a *context* map with the attributes of the of the user groups (e.g. the branch from organizational chart where the users belongs to). This means the tool is creating a list of users enabled to get access to the files considered in a given information sharing operation. This tool can be configured to adjust this type of decision.

3.2. Risk Assessment

The risk assessment module considers sensitive criteria (Sc) to calculate a risk score used to define mitigation actions for each file processed by SecFilter. In practice, this module receives the content and context criteria identified by the previous module and performs a comparison with the criteria established by the organization as sensible. In order to perform this task, two types of risk assessment policies are considered in this paper:

- Scoring criterion policy.

  This policy establishes a single criterion of sensitivity (Sc), which determines the risk score (RS) of each file:

  $$RS(d_j) = Sc \text{ where } 0 < Sc \leq 1$$

- Scoring criteria policy.

  In this policy, SecFilter is configured to consider multiple sensitive criterion. The module estimates a score having a value between 0 and 1 for each sensitive criterion (Sc_i) and defines a weight for each criterion (Wc_i) in the policy. Both values (Sc_i and Wc_i) are input parameters required to calculate the risk level (RS) of a document (d_j). The risk level of a document, using n sensitive criteria, is determined by equation 2.

  $$RS(d_j) = \sum_{i=1}^{n}(Sc_i * Wc_i) \quad (2)$$

  where

  $$\sum_{i=1}^{n} Wc_i = 1 \text{ and } 0 < Sc_1, Sc_2, Sc_3 ... Sc_n \leq 1 \quad (3)$$

From the above formulas it is inferred that the weight for each criterion, as well as the level of sensitivity, may be different, but the sum of the weights for each criterion must be equal to one and each Sc value must be between 0 and 1 (see equation 3). Both constraints allow the resulting value of RS to always be a value between 0 and 1, which allows associating this value to a range of values called *risk levels*, which are associated with a given action.

The risk assessment module manages risk score ranges. In this paper, three types of ranges are considered, *low* from 0 to 0.33, *medium* from 0.34 to 0.66 and *high* from 0.67 to 1, but organizations can adjust these ranges to be considered by SecFilter depending on a given ISO. The final tasks performed by this module are to determine the risk level (RL) for each RS (risk score) calculated per policy, as well as to create a map including the ID file (file identifier) and the RL identified by SecFilter. This map is sent forward to the next stage in the SecFilter pipeline.

3.3. Security and mitigation manager

The security manager defines the mitigation actions for securing each file depending on the maps created by the risk assessment module.

The security manager determines as many mitigation actions as the number of risk levels (low, medium, and high) managed by the risk assessment module. The security manager in SecFilter has the capacity to increase the size of the data encryption key [65] when the risk level increases. Table 1 shows the default actions for each risk level.

Table 1: Mitigation action for each risk level.

Risk level	Risk Range	Mitigation
Low	$0 < RL \leq 0.33$	K=128bits and Access Control
Medium	$0.34 \leq RL \leq 0.66$	K=192 bits, Access Control and Integrity
High	$0.67 \leq RL \leq 1$	K=256 bits, Access Control, Integrity, and Signature

The security manager uses a multi-level cryptographic engine, conceptually based on a realization of digital envelopes that ensure not only confidentiality and access control over the data, but also integrity and authenticity by means of digital signatures [13]. Originally, the operations are generically created for either generating (policy generation, 2 encryption layers, hashing and signature generation) or opening (decryption key generation from attributes, 2 decryption layers, hashing and signature verification) the digital envelopes. In SecFilter, those operations are customized to the owner requirements. The impact on the encryption/decryption cost under the digital envelope concept is reduced as the most expensive procedure (ABE encryption) is performed over symmetric keys, not over files. Usually, the keys are no longer than 256 bits, compared to data size, which can be in the order of mega or gigabits.

Conceptually, data are symmetrically encrypted by using a secure session key (k), whose size depends on the risk level in the map received as input parameter. The file (F) indicated in the map is encrypted using AES with the chosen size of k. This procedure produces two files (k and an encrypted version $|\widehat{F}|$ of F). With CP-ABE, an encrypted version $\widehat{k}$ of k is created, using policies from attributes included in the map received from the previous module in the SecFilter pipeline.

In order to provide integrity, the security engine creates a hash ($\widehat{H}_{|F|}$) from F and then signs that hash ($\widehat{Sig_H}$), thus also providing authentication and non-repudiation. The digital signature is done using a specific key-pair $\{S_k,\ P_k\}$, being S_k the private key for signing and P_k the public one for verification. This key-pair can be either created for the whole organization, for the SecFilter instance, for the data owner, or for the file being processed. The choice will depend on how the organization deploys SecFilter. In this paper, we use a key-pair for each data owner in the organization.

The file features can also be considered in the mitigation actions, as this module is configurable. In this paper, only the size of the key is considered in mitigation actions, whereas confidentiality (encryption), access control (attribute-based encryption), and integrity (digital signatures) are performed by default. However, they can be disabled on-the-fly and on-demand depending on the SecFilter configuration. For instance, besides the key size, SecFilter could only consider confidentiality and access control for low risk level, confidentiality, access control and integrity for medium, and confidentiality, access control, integrity, and authentication/non-repudiation for a high-risk level.

The digital envelopes DE_F, produced by the security engine per each processed file, concatenate $\widehat{H}_{|F|}$, $\widehat{Sig}_H$, $|\widehat{F}|$ and $\widehat{k}$ into a single file. Algorithm 3 describes the encryption procedure performed by SecFilter. When the DE_Fs has been generated, SecFilter can send it to either the cloud or other repository by using the corresponding write/upload operations.

Algorithm 3 Encryption process in the security manager

Require: File F, Key-pair $\{S_k, P_k\}$ of F's owner, Risk level l, Policy p

%Digital signature generation
$\widehat{H}_{|F|} \leftarrow$ getHash(F)
$\widehat{Sig}_H \leftarrow$ sign($\widehat{H}_{|F|}$,S_k)
%File encryption with AES
$k \leftarrow$ secureKeyGenAES(l)
$|\widehat{F}| \leftarrow$ AES.encryptFile(F, k)
% Key Encryption with CP-ABE
$\widehat{k} \leftarrow$ CP-ABE.encrypt(k, p)
$DE_F \leftarrow |\widehat{F}| \;||\; \widehat{k} \;||\; \widehat{H}_{|F|} \;||\; \widehat{Sig}_H \;||\; P_k$
return DE_F

In the download operations performed by user u, the DE_F being retrieved from the cloud or local repository is split in its components, to perform one of the following operations:

1. Access to F contents, in plain text (confidentiality and access control). The user u has a set of attributes, assigned by the system prior to perform the downloading operations. This is done for example, during a register operation in the system. With those attributes, the system also delivers to the user a unique decryption key k_u. With that key, the user recovers the data encryption key, k = CP-ABE.decrypt($\widehat{k}$,k_u). In this process, k is used to decrypt the data and to recover the original file F = AES.decryptFile($|\widehat{F}|$, k), only if the attributes that generate k_u satisfy the policy p used during the encryption phase. This verification is ensured by the internal algorithms of CP-ABE.

2. Integrity checks (confidentiality, access control and integrity). This is achieved by verifying the hash provided in DE_F. It is accomplished by comparing the hash of the decrypted file $H_{|F|}$ = getHash(F) with the hash $\widehat{H}_{|F|}$ in DE_F.

3. Digital signatures (confidentiality, access control, integrity and authentication/non-repudiation). The public key P_k in the digital envelope is used to decrypt $\widehat{Sig}_H$, and to recover the original $\widehat{H}_{|F|}$ computed during the encryption phase. The decrypted data in F are considered as authentic if, and only if, the re-computed hash of the decrypted file, $H_{|F|}$, $H_{|F|}$, is equal to $\widehat{H}_{|F|}$.

At this point, we remark the advantages of digital envelopes over attribute-based encryption. Unlike traditional public key encryption, in the ABE encryption scheme, the public key of the receivers is not required in advance, but encrypts to all those whose attributes (Att) satisfy the access policy (p) to provide confidentiality and fine-grained access control over the symmetric key.

Although we conceptually use the AES4SeC approach [13] for creating the digital envelopes, the deployment in SecFilter is different than this approach. The encryption engine is actually encapsulated into virtual containers to build a processing pipeline that allows creating the following possible configurations:

1. Multi-security level. Encryption can be done with a strength of 128, 192, or 256 bits. It is ensured that CP-ABE, hashing and the digital signatures are compliant with the standardized security levels supported by AES. By itself, the deployment of CP-ABE and pairing based digital signatures is complex because it involves selecting an elliptic curve, pairing setting, groups size, attributes management, among others. In SecFilter, it is transparently addressed by the security manager.

2. Security services. SecFilter can provide three possibilities of security services over the files: {*confidentiality, fine-grained access control*}, {*confidentiality, fine-grained access control, integrity, authentication*}, and {*integrity, authentication*}. In the last option, the digital envelope is not used, only digital signatures are computed.

4. Security Filter prototype

To assess our proposal, a security filter prototype was developed as a pipeline of five chained virtual containers: interception (Input); criteria veri-

Table 2: Characteristics of the instances in the private cloud.

Image	**# of instances**	**Cores**	**RAM (GB)**	**HD (GB)**
Storage	5	4	6	80
Metadata	1	2	2	100

fication; risk assessment; security engine; and delivery (Output). For evaluation purpose, a bot (*client*) was also implemented to produce file sharing workloads. This client produces sharing operations (with valid credentials) and sends files to a folder monitored by SecFilter, which assumes that this workload comes from real users. SecFilter was configured to deliver the secured files to a fault-tolerant, multi-cloud storage, and content delivery service called SkyCDS [66], which also includes Pub/Sub operations that enable users to create sharing workflows automatically. SkyCDS was implemented by using cloud images deployed on a private cloud built with Openstack Mitaka.

It is important to note that SkyCDS can be replaced with other storage system or sharing application (E.g. dropbox, RapidShare, etc). We have chosen SkyCDS for evaluation purposes because we can install this service in a private cloud. Table 2 describes the features of the instances deployed in the private cloud.

In the dataflow performed in the container pipeline implemented by SecFilter prototype, the client bot sends files to a folder called *in bucket*. The interception module takes files from this folder and proceeds to send them to the pipeline. The documents are processed for the assessment of risks and the files secured, in the form of digital envelopes, are delivered to an *out bucket*. The secured data are extracted by a sync tool of SkyCDS that sends them to a fault-tolerant multi-cloud platform. As it can be seen, the administrator only requires to configure SecFilter with the paths of the *in* and *out* buckets; as a result, SecFilter can build a dataflow to provide upload and download operations. Notice that this type of flow includes an analysis of sensitive topics criterion, which can be omitted depending on the organization needs or types of files managed in the organization.

4.1. Task Parallelism in SecFilter

The modularity of SecFilter design and its implementation using containers enable organizations to create parallelism patterns to enhance the processing of large files and large volumes of files, which will be required in organizational scenarios. As a proof of concept of this design feature, the module for criteria verification was built using a farm of n containers, each with 6 cores and 12GB RAM. A distributed processing pipeline was built using this farm in parallel for criteria verification and data analytic, showing that the efficiency of those steps was improved, in terms of service times, without breaking the SecFilter dataflows. The processing pattern was built by using Docker[2]). The containers are chained in pipelines by using network and memory I/O interfaces [37]. This pipeline is depicted in Figure 3.

The pipeline for the topic detection processing includes a set of sub-modules such as: module for documents loading, module for topic detection and module for results integration. These modules were deployed on the private cloud. The pipeline is considered by SecFilter as a single stage that receives data from the first stage (interception) and sends results forward to the risk analysis stage.

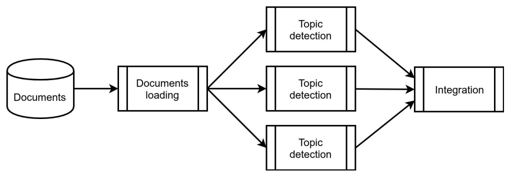


Figure 3: Pipeline for topic detection including the parallel processing stage.

Note that this type of parallel patterns can be deployed at each stage of SecFilter.

4.2. Graph Visualization Tool

When SecFilter is configured to verify the presence of sensitive topics within the file content, a topic graph is built to describe, in a visual manner, each topic found in the analyzed set of documents. A Graph Visualization Tool (GVT) has been also created to enable the decision-makers to discover topics and to create a *lexicon* of sensitive terms for a given organization by using the discovered terms. Figure 4 show examples of map graphs of document-per-topic.

This tool also shows the volume of consumption/production of the discovered topics in the sharing operations, as well as a big picture of the sharing operations. No personal information is exposed

[2] www.docker.com

in graphs, as SecFilter secures the information required to display graphs using digital envelopes. As a result, only administrators can provide this tool with the information to create graphs. SecFilter also includes a GUI for administrators to define the number of criteria and the weight for each criterion to be analyzed by SecFilter.

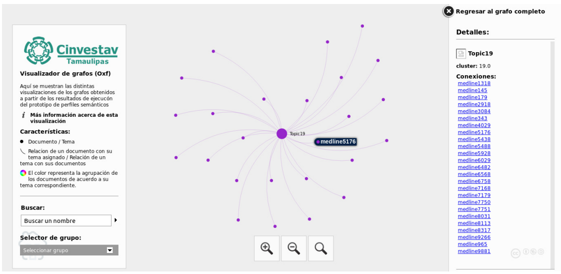


Figure 4: Example of graph of maps (Documents-per-Topic) created by GVT

5. Performance evaluation

In this section, we describe the experiments run to test the system, the metrics used to evaluate our proposal, the results discussion and the comparisons against other approaches.

5.1. Methodology

The experimental evaluation of SecFilter was conducted in three stages: In the first one, a performance evaluation for the security filter module was performed; In the second one, a case study based on a set of documents, including MedLine corpus, was tested. In the last one, satellite images provide by a space agency (European Space Agency or ESA) were also processed by SecFilter.

The following metrics were used to evaluate the experiments:

- *Service time (st)*: Time spent in the Secfilter pipeline, which can be calculated as the sum of the time spent by each module (m) in a pipeline of SecFilter. This metric is also considered as a processing time.

$$st = \sum_{i=1}^{n} st(m_i)$$

- *Response time (rt)*: This metric represents the service experience observed by end-users, which includes the service time (st) plus the time spent during the interception and retrieval/delivery of files from/to the cloud (T_{ta}).

$$rt = st + T_{ta}$$

- *Throughput (th)*: The median amount of the data secured by SecFilter, in megabytes, per time unit (MB/s).

$$th = \sum_{i=1}^{n} F_{size_i}/rt$$

5.2. Performance evaluation of SecFilter modules

A fine tuning of parameters of the procedure for the verification of sensitive topics and security/mitigation modules of SecFilter was performed. We consider that this will help decision-makers to choose the most suitable configuration depending on the organization needs. Interception and delivery modules were not tuned as these modules depend on the sharing application and cloud services respectability, whereas the risk assessment functionality depends on the verification of sensitive topics and security/mitigation modules of SecFilter.

5.2.1. Criteria verification; data mining and text parser fine-tuning

The percentage of documents mapped with a topic is critical for this module to process a data source (e.g. either a catalog or dataset). Minimum and maximum values for word frequencies are used to eliminate the relevance of words in lower and higher extremes of the dataset's vocabulary (i.e., the infrequent and very frequent words in the vocabulary). *Minimal Frequency (FMIN)* is used to eliminate rare words from the vocabulary. Thresholds were defined oscillating in the range [5, 100]. *Maximal Frequency (FMAX)* is used to eliminate words with high occurrence in the vocabulary. Thresholds were defined oscillating in the range [0.05, 1.0].

The experiments were performed using a dataset of 100,000 abstracts from *MedLine* corpus[3]; the median of words for abstracts was 298. From the dataset, we obtained 100 samples selected in a random fashion, the number of documents per sample increased by 500 in a range of [3000, 10000].

[3] `https://www.ncbi.nlm.nih.gov/pubmed/`

Each sample of documents was processed by the topic detection module using the default value for maximum frequency and varying the minimum frequency, as well as using the default value for minimum frequency and varying the maximum frequency. In more detail, the minimum frequency was increased by 25 units starting with the default value 7; As a result, all samples were tested by using values such as 7, 25, 50, 75, 100. The maximum frequency was increased by 0.2 units starting with a default value 0.084; As a result, the samples were tested with values 0.084, 0.1, 0.3, 0.5, 0.7, and 0.9.

Each sample of documents from MedLine was processed 30 times, one per combination of values of minimum and maximum frequencies; As a result, 3000 experiments were performed for the 100 samples considered in this fine-tuning evaluation. Each sample was evaluated once per each combination of frequencies, as the topic analyzer produces deterministic maps of documents.

An analytic processing pipeline in SecFilter was deployed on a cluster by using parallel patterns in which one master container receives documents (*documents loading*) and distributes this workload to four slave containers (Each slave has inside another farm with three containers for *topic detection*), which deliver results to the container for integration and consolidation of results (*results integration*). This container sends the consolidated results to the next stage in the SecFilter pipeline.

In each experiment we obtained the number of identified topics and the number of documents with topic, capturing the maximum number of documents mapped with topic per experiment, which are shown in Figure 5.

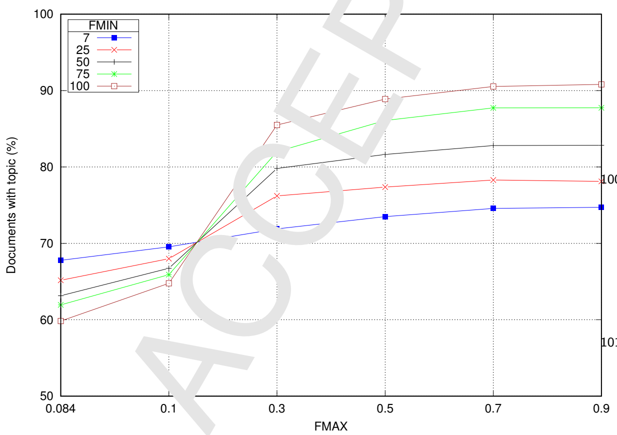


Figure 5: Tuning evaluation results.

As it can be seen, the configuration $FMIN = 7$ obtained a minor variance in the percentage of documents mapped with topic and the percentage of documents mapped with topic converges all values of $FMIN$ for $FMAX = 0.152$. With $FMAX \geq 0.3$, the percentage of documents with topic increases proportionally to $FMIN$. In scenarios of $FMAX = 0.152$ with $FMAX \leq 0.1$, the percentage of documents mapped with topic decreases inversely proportional to $FMIN$. From $FMAX \geq 0.5$ a stable behavior was observed for all values of $FMIN$. An INOVA study showed that each result showed in Figure 5 has a 5% of error margin.

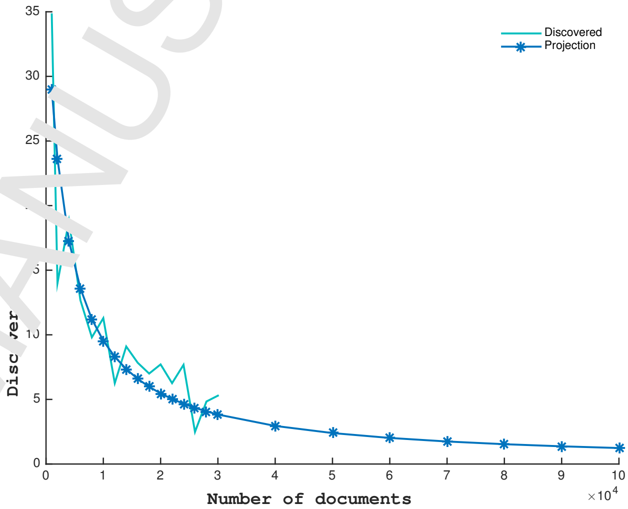


Figure 6: Topic discovering in MedLine corpus.

n

Figure 6 shows the mean amount of topics discovered by the analytic module for the samples of MedLine corpus, as well as a projection for large number of documents. As it can be seen, most of the topics are discovered soon in the curve. Thus, analyzing ten thousand documents could capture the most of topics from the set of documents[4].

5.2.2. Security and mitigation module tuning

This module was configured with the setup parameters reported in [13], using those that lead to the best performance and have minimum memory requirements. The setup included selection of the elliptic curve (F), pairing type (asymmetric or

[4] A similar effect to the process of building a dictionary is observed in this experiment: the more words are discovered soon and few words are added each year

Type 3), and groups size compliant with the key sizes in Table 1. This configuration was then evaluated by deploying the security module in a container, to determine the cost (response time) of confidentiality, as well as access control and integrity, separately from the entire SecFilter process.

The confidentiality feature was tested with different size keys, the access control was evaluated with different number of attributes and encryption policies, and integrity was tested over files of different sizes. The security level in the tests for access control and integrity checks were always consistent with the security level indicated for confidentiality. The same set of documents was used for all the test, varying the size of the shared files (1Kb,1MB,10MB and 100MB were tested).

Figure 7 shows the evaluation of the security module in SecFilter. Two configurations for the pipelines of the encryption engine were tested based on confidentiality and access control (C-CA): The first one considers confidentiality, access control and integrity for medium risk level (C-CA-I), whereas the second one considers confidentiality, access control, integrity, and authentication/non-repudiation for a high-risk level (C-CA-I-A). For each configuration we used encryption policies of 1, 2, 3 and 4 attributes. In all the tests, a 10MB file was used.

As it can be seen, the more the attributes used in encryption operation and the more the risk level (key size), the more the cost observed by end-users in the configurations. The overhead in response times produced by integrity and non-repudiation features does not grow proportionally with the size of AES key, but in exponential manner as shown in that Figure. Performance degradation of low security level was 10%, medium security level was 95% and 112% was observed when using a high security level.

The results showed in Figure 7 could guide administrators to identify the costs of a comprehensive set of configurations of the multi-security engine of SecFilter [5]

5.3. A case study based on repositories of documents

In order to evaluate SecFilter in a real-world scenario, we conducted a case study based on ensuring a set of documents from MedLine repository by using SecFilter and SkyCDS as Cloud storage service.

[5]A more detailed performance evaluation is available in [67].

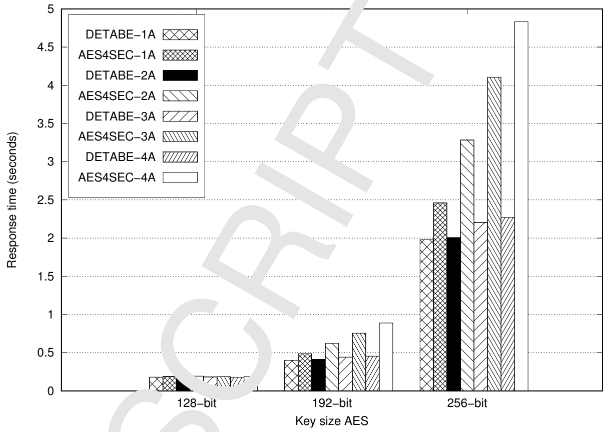


Figure 7: Response time for different configurations of the security module in SecFilter, different risk levels (key size) and attributes number.

5.3.1. Configurations and experiments

For the experiments, all the modules in SecFilter were set up as in the previous fine-tuning evaluation. A scoring criteria policy was used for the verification of sensitive items. This policy considers the topic sensitivity (ST) of the documents' content as *Criterion 1*, whereas the context sensitivity related to the hierarchical level (SH) of document owners is considered as *Criterion 2*.

The risk assessment module was configured to manage low, medium and high-risk levels, whereas the security module was configured with a map of *mitigation actions* that considers varying the key size depending on the risk level (low:128, medium:192 and high:256) as well as the on-the-fly selection of the encryption attributes for each processed file.

With this configuration, the risk level (RL) of a document (d_i) for the scoring criteria policy is determined by equation 4.

$$RL(d_i) = (ST * WT) + (SH * WH) \qquad (4)$$

The $MedLine$ corpus and an organizational chart of 4 levels were used in this study case. A sample of 10,323 documents were extracted randomly from this corpus to conduct the experimentation. The topic detection tool was executed to determine the maximum (54 topics) and mean (13 topics) found in this sample. Two lexicons were created with these group of topics to conduct two types of experiments.

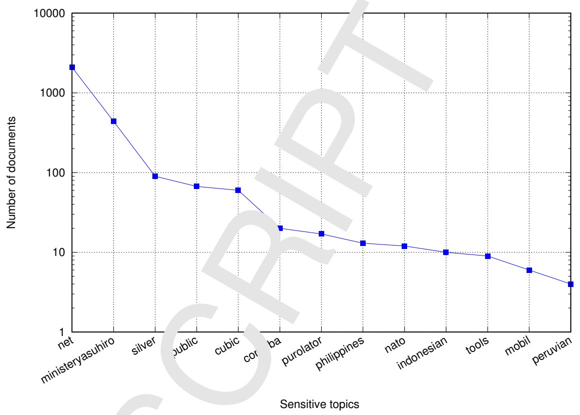
SecFilter was configured with three different configurations to assess the risk of each document of the sample varying the weight of the criteria considered by SecFilter: i) *WT = 0.5, WH = 0.5*, so that the user/organization considers equal security priority for the place, in a hierarchical organizational chart (WH), of the users involved in a sharing operation and the terms of the lexicon of sensitive topics (HT) found in the documents; ii) *WT = 0, WH = 1*, so that the organization only considers as priority the organizational hierarchy, whereas the topics of the lexicon are not considered in the risk assessment; iii) *WT = 1, WH = 0*, so that organizations choose considering only sensitive topics in the risk assessment of documents.

5.3.2. Analyzing results of lexicon scenarios

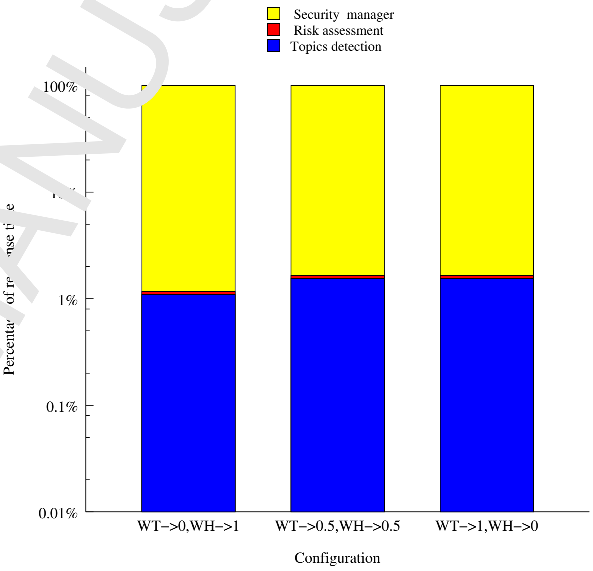
This section presents the results in two scenarios. In the first one, SecFilter manages a lexicon including 13 sensitive topics, in the second one 54 topics are considered.

Figure 8 shows the documents mapped with a topic when SecFilter is configured with a lexicon of 13 sensitive topics (8a) as well as the response time portion of each module of SecFilter (8b) for the three configurations. As expected, and pointed out in fine tuning evaluation, a large portion of documents is mapped to few topics. The performance costs of the filters of the Secfilter pipeline increases in one magnitude order: risk assessment represents 1%, topic discovering and criteria verification, whereas the securing of files represents 99%. This means that the costs of analyzing content and context of file, as well as assessment risks of them are not representative for performance issues in comparison to the basic tasks of securing data.

Figure 9 shows the decisions taken and actions made by SecFilter in terms of documents encryption using 128 (low risk), 192 (medium) and 256 bits (high) respectively (9a), as well as the performance impact on the throughput produced by actions made (9b). Interesting effects can be observed in 9a: i) A large document portion is labeled with low risk when organizations, in the risk assessment, only consider the context of the documents that is defined by the place of the users involved in sharing operations (WH=1;WT=0). ii) The more the weight assigned to the content of documents criterion (WH=0;WT=0.5 and WH=0;WT=1), the more the portion of documents mapped with medium and high risk levels and ensured by SecFilter with 192 and 256 key sizes. Figure 9a, shows, as expected, that, in that case,



(a) Number of documents per topic discovered.

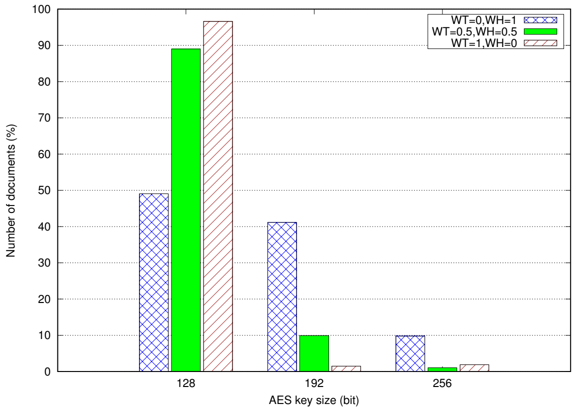


(b) Response time percentage of SecFilter modules.

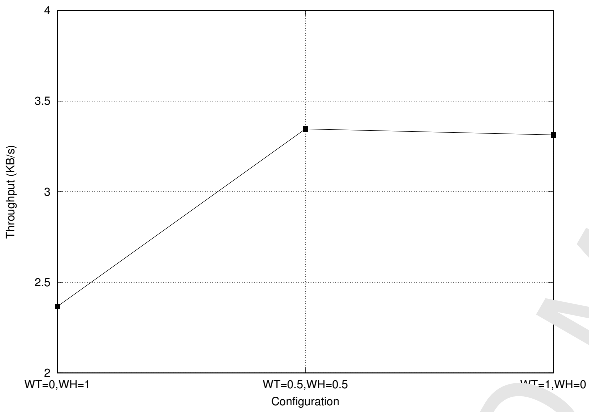
Figure 8: Lexicon of 13 sensitive topics and SecFilter performance per module.

the majority of documents demand the 192-bit key size, which implies more work, thus affecting the throughput of SecFilter.

Figures 10a and b show similar behavior to results showed in Figures 8a and b, whereas Figure 10c reaffirms the results showed in Figure 9a about the criteria used in SecFilter: analyzing context produces an increment of documents mapped with low risk, whereas analyzing contents produces an increment in the usage of large key sizes. The more the criteria used in risk assessment, the bigger the increment in the number of documents mapped with medium and high risks. Moreover, it also

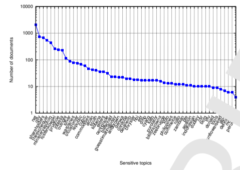


(a) Percentage of documents secured for each security level (AES).

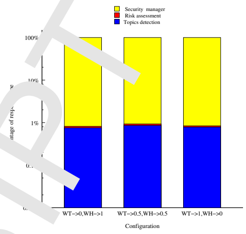


(b) Throughput of SecFilter per configuration.

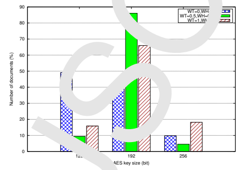
Figure 9: Performance of SecFilter when managing 13 sensitive topic scenario.



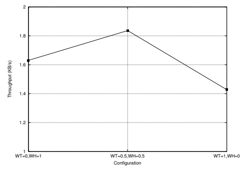
(a) Number of documents per discovered topic



(b) Response time percentage of SecFilter modules.



(c) Percentage of documents secured for each security level (AES).



(d) Throughput of SecFilter per configuration.

Figure 10: Scenario when SecFilter manages a lexicon of 54 sensitive topics.

shows that the usage of keys with larger size (192 and 256) increases when the size of the lexicon of sensitive topics also increases, which means that more risky situations were discovered by SecFilter in this scenario than in the previous one. Figure 10d shows an increment in the throughput produced by SecFilter when managing a set of criterion and several sensitive topics in the lexicon.

5.4. Performance comparison results

In this section we show a comparison of the SecFilter performance with the performance of a solution where users are responsible of encrypting their files, using a CP-ABE solution with static key sizes ($Sec128$ and $Sec256$) and a configuration where a portion of users do not perform the encryption of 40% of their files before sending them to the cloud ($Sec128(60-40)$). In these experiments, the configurations under study encrypt and secure ten thousand of papers (associated to abstracts of MedLine corpus) arriving to a folder in a computer of a real user, which are downloaded by other users through the cloud. Figure 11 shows the average response time observed by end-users when sharing files of a folder (Sharing + Upload). As expected, $Sec256$ represents the most expensive solution for organizations, whereas the configuration where users encrypt only 60% of files (chosen in random manner by the bot) produced a similar performance to $Sec256$. This means that, in terms of performance, installing a filter, even by using a fix size key (128), is feasible in comparison with completely relegating the security to end-users. As it can also be seen, SecFilter produces an acceptable performance when considering criteria in risk assessment (14% more than $Sec128$). This performance can be improved in large volumes of data by using parallel pattern creating the clustering of containers (see prototype and fine-tuning sections).

5.5. Analyzing SecFilter performance when using parallel patterns

In order to improve the performance of the security manager (the critical stage in SecFilter), we developed a parallel pattern to process the contents

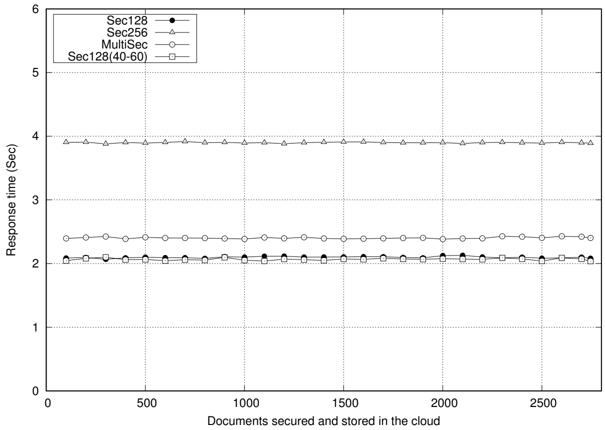


Figure 11: SecFilter and AES comparison when sending data to the cloud

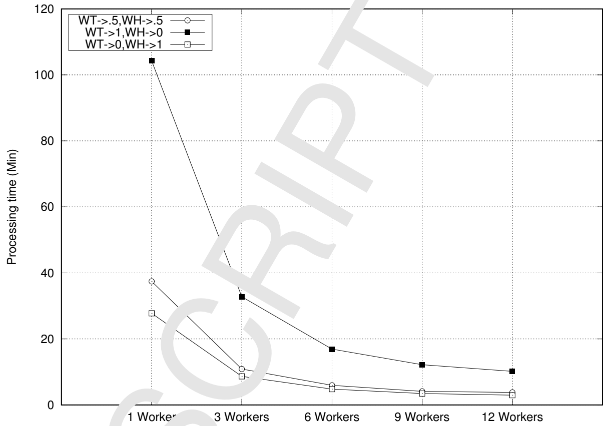


Figure 12: Response time produced by SecFilter configurations when using task parallel patterns for the security manager.

in a concurrent manner (see Task Parallelism in section 4.1).

Data task parallel patterns have been included in the SecFilter container and performed a set of experiments by varying the number of workers (1,3,6,9 and 12) in a computer of 12 physical cores. A set of files from the MedLine corpus (2745 files) was processed by varying the sensibility priority degree ($WTandWH$) in the same way performed in previous experiments.

Figure 12 shows the performance of SecFilter when processing the set of files placed by a bot in the data source (vertical axis) produced by a pattern including different number of workers (horizontal axis). As expected, the performance of SecFilter was improved when increasing the number of workers. The SecFilter performance improved up 90% when the pattern deploying 12 workers on the container in comparison with the original version of the security manager previously evaluated. The behavior is similar to the one previously described: more files are secured with large keys when the content has more weight than the context (See WT->1, WH->0), less files are secured with large keys when the context has more weight than the content (See WT->0,WH->1). Again and as it is expected, the combination of context and content analysis produces secured files with different security key sizes (See WT->0,5,WH->0,5).

Figure 13 shows the processing time (vertical axis) produced by SecFilter configurations and AES configurations when using fixed security key sizes (horizontal axis). SecFilter configurations for different sensibility weights were tested. In

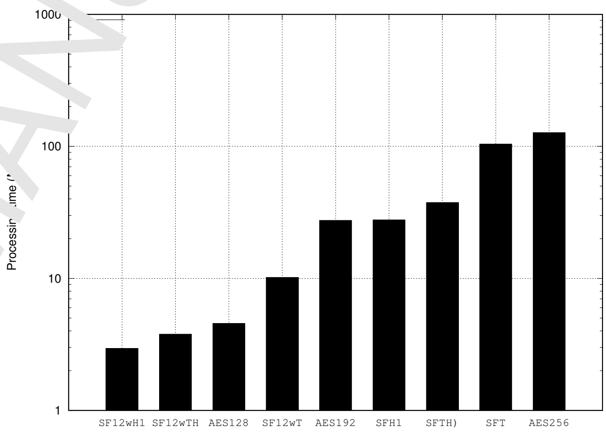


Figure 13: Response time produced by SecFilter configurations when using fixed security key sizes.

the SFT configuration, the content (topics) is important (WT=1 and WH=0), whereas the context is not (Hierarchy). In the SFH configuration (WT=0790 and WH=1), whereas SFTH (WT=0,5 and WH=0,5). When these configurations deploying task parallel patterns with 12 workers, the names of configurations contain the 12w suffix (SFT12w, SFH12w and SFTH12w). As it can be seen, the versions of SecFilter using parallel patterns (see configuration SF with the 12w suffix) produce a better performance than the solution using only keys of 128 bit size. In fact, the SFTH12w configuration where an equal weight is assigned to content and context produces a 17,15% improvement, whereas 35,45% of improvement is achieved by the configuration where the context is the priority (SFH12w). The configuration of SecFilter based only on content analysis cannot produce a better performance than the solution only using a 128 key but produced a significant improvement in comparison with the solutions only using a 192 and 256 key sizes.

5.6. Analyzing SecFilter performance when ensuring satellite images

A subset of 20 satellite images of the SMOS mission proportioned by the European Space Agency (ESA) was secured by SecFilter configurations. Experiments varying the number of workers in the SecFilter patterns (1,3,6,9 and 12) were performed to process a 731MB repository of images with a mean size of 40MB. The configurations of SecFilter considered in these experiments are SFH configuration where only the context of the images is taken into account (WT=0 and WH=1) as well as configurations using only 128, 192 and 256 key sizes (*SF128, SF192 and SF256*) to process all the images in the repository.

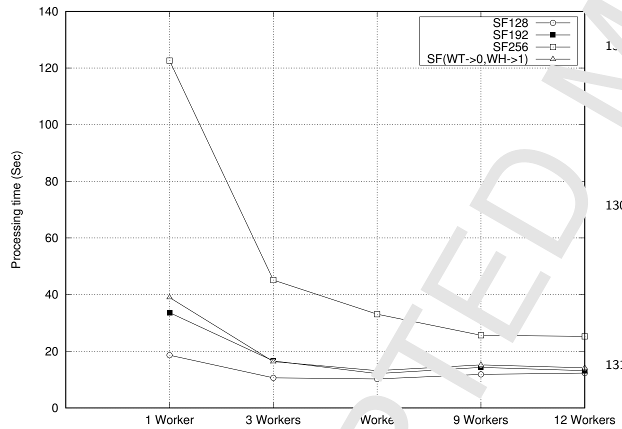


Figure 14: Response time produced by SecFilter configurations when processing satellite images by using fixed security key sizes

Figure 14 shows the processing time (vertical axis) produced by the SecFilter configurations when varying the number of workers (horizontal axis). As it can be seen, SecFilter can be used either only as multi-security scheme (without considering the content nor context of files) to improve the performance of the security solutions or also including the context/content sensible criteria detection. The cost of using different types of security levels, depending on the content and context, shows that, the application of SecFilter to real-world applications is not only feasible, but also efficient, when using the parallel patterns created because of the modularity of the SecFilter architecture based on containers.

6. Conclusions and future work

In this paper, we focused our attention on events where sensitive information is unintentionally exposed and/or sent to the cloud without encryption by insiders that even were previously informed about cloud risks. Such an effect is called *information privacy paradox* in the literature and affects mainly to organizations in scenarios of information sharing.

In this context, we proposed combining smart and reactive solutions with encryption and security policy schemes for organizations to face up this type of challenge instead only using an encryption solution. We presented in this paper, the design, development and implementation of SecFilter, a security filter that enables organizations to mitigate the effects of the information privacy paradox by implementing security policies for information sharing scenarios. SecFilter automatically performs the following tasks: a) intercepts files before sending them to the cloud; b) searches for sensitive criteria in the context and content of the intercepted files by using mining techniques; c) calculates the risk level for each identified criterion; d) assigns a security level to each file based on the detected risk in its content and context; e) encrypts each file by using a multi-level security engine, based on digital envelopes from symmetric encryption, attribute-based encryption and digital signatures to guarantee the security services of confidentiality, integrity and authentication on each file at the same time that access control mechanisms are enforced before sending the secured file versions to cloud storage. The modularity of the design of this security filter, based on clusters of virtual containers, enables organizations to build solutions facing up scenarios of management of large volume of data.

In order to show the feasibility of implementing this security filter, a prototype of SecFilter was implemented for a real-world file sharing application and deployed on a private cloud. Fine-tuning of SecFilter components is described and a case study was conducted based on document sharing of well-known repositories (Medline). The experimental evaluation revealed the following results: i) the components of SecFilter do not produce a significant impact over the service experience of the end-users as storage and security represent the major costs of securing the files; ii) when organization chooses a fixed configuration, all the files are encrypted before sending them to the cloud and there is no a significant impact over performance in comparison when users sporadically ignore to encrypt files (scenarios where users ignoring the encryption of 40% of files were tested); iii) when more criterion are analyzed and verified by SecFilter more risky situations are discovered and more mitigation actions are performed, which are only 14% more expensive than using a fixed security level and the encryption of a portion of files. In SecFilter, clusters of containers producing task parallel reduce response times of the processes performed by SecFilter. Experimental evaluation revealed that the parallel patterns improve the performance of SecFilter configurations up 90% when increasing the number of workers launched in the SecFilter container. The SecFilter configurations using parallel patterns produced a better performance than solution using only keys of 128 bit size. In fact, configurations of SecFilters where equal sensibility weight is assigned to content and context produce 17,15% of improvement in comparison with solutions using only keys of 128 bits, whereas 35,45 % of improvement is achieved by the configuration where the context is the priority in the sensibility criteria. The configurations of SecFilter based only on the identification of sensible criteria in the content of files can not produce a better performance than the solution only using a 128 key but it produced a significant improvement in comparison with the solutions using only a 192 and 256 key sizes. The behavior observed when SecFilter ensuring files by processing the content of documents was also observed when it ensured satellite images by processing the context of these files, which showed the flexibility of applying this solution to different domains.

In this paper, parallel patterns based on distribution of tasks were developed within the virtual container of SecFilter as a scheme to improve the performance of analytics and encryption procedures. Nevertheless, we are considering that the AES software included in SecFilter could be replaced with improved realizations based on AES-NI in the near future. In such as a case, AES-NI dependencies and libraries could be encapsulated into the virtual container used to deploy SecFilter on the end-users; as a result, extra configurations and troubleshooting could be avoided for end-users. We also will work on testing other content analysis tools for different types of file formats for the first stage of SecFilter as this stage only considers text mining as a sensible content identification by now.

The current version of SecFilter prototype supports the risk assessment in the context of any type of file, but it only implements the analysis of content in documents/texts by now. A quite interesting opportunity area for SecFilter is analyzing the content (not only the context) of other types of contents (not only documents), which is feasible because of the modular architecture of SecFilter.

In this context, we think it is possible for users to change the content analytic module based on mining text currently available in the SecFilter prototype, for instance, by a voice and signal analyzer. SecFilter could find sensitive words and/or topics associated to the words in audios. This could be an interesting research line for our group to explore in the future.

Acknowledgment

This work has been partially supported by the Spanish "Ministerio de Economia y Competitividad" under the project grant TIN2016-79637-P "Towards Unification of HPC and Big Data paradigms".

References

[1] K. Kasemsap, Mastering intelligent decision support systems in enterprise information management, in: Web data mining and the development of knowledge-based decision support systems, IGI Global, 2017, pp. 35–56.

[2] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, J. Molina, Controlling data in the cloud: Outsourcing computation without outsourcing control, in: Proceedings of the 2009 ACM Workshop on Cloud Computing Security, CCSW '09, ACM, 2009, pp. 85–90.

[3] R. H. Sloan, R. Warner, Unauthorized Access: The Crisis in Online Privacy and Security, 1st Edition, CRC Press, Inc., Boca Raton, FL, USA, 2013.

[4] . h.-c.-o.-a.-d.-c.-o. Data Center Frontier. January 25, The cost of data center downtime, accessed 16.01.18.

[5] C. Lee, C. C. Lee, S. Kim, Understanding information security stress: Focusing on the type of information security compliance activity, Computers and Security 59 (2016) 60 – 70.

[6] Y. Cherdantseva, P. Burnap, A. Blyth, P. Eden, K. Jones, H. Soulsby, K. Stoddart, A review of cyber security risk assessment methods for scada systems, Computers and Security 56 (2016) 1 – 27.

[7] H. g. PWC, 2015 information security breaches survey.

[8] IDG, 2017 u.s. state of cybercrime executive summary, accessed 16.01.18.

[9] RSA, 2016: The current state of cybercrime.

[10] R. D. I. Gantz J, E. Report, Extracting value from chaos.

[11] R. D. I. Gantz J, E. Report, The digital universe in 2020: big data, bigger digital shadows, and biggest growth in the far east.

[12] B. Waters, Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization, in: D. Catalano, N. Fazio, R. Gennaro, A. Nicolosi (Eds.), Public Key Cryptography – PKC 2011, Springer Berlin Heidelberg, 2011, pp. 53–70.

[13] M. Morales-Sandoval, J. L. Gonzalez-Compean, A. Diaz-Perez, V. J. Sosa-Sosa, A pairing-based cryptographic approach for data security in the cloud, International Journal of Information Security.

[14] J. Li, Y. Zhang, X. Chen, Y. Xiang, Secure attribute-based data sharing for resource-limited users in cloud computing, Computers and Security 72 (2018) 1 – 12.

[15] W. Kearney, H. Kruger, Can perceptual differences account for enigmatic information security behaviour in an organisation?, Computers and Security 61 (2016) 46 – 58.

[16] S. Kokolakis, Privacy attitudes and privacy behaviour: A review of current research on the privacy paradox phenomenon, Computers and Security 64 (2017) 122 – 134.

[17] S. V. Flowerday, T. Tuyikeze, Information security policy development and implementation: The what, how and who, Computers and Security 61 (2016) 169 – 183.

[18] S. Weigert, M. Hiltunen, C. Fetzer, Mining large distributed log data in near real time, in: Managing Large-scale Systems via the Analysis of System Logs and the Application of Machine Learning Techniques, SLAML '11, ACM, New York, NY, USA, 2011, pp. 5:1–5:8. doi:10.1145/2038633.2038638.
URL http://doi.acm.org/10.1145/2038633.2038638

[19] J. Besson, C. Robardet, J.-F. Boulicaut, Mining formal concepts with a bounded number of exceptions from transactional data, in: B. Goethals, A. Siebes (Eds.), Knowledge Discovery in Inductive Databases, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 33–45.

[20] F. P. Miller, A. F. Vandome, J. McBrewster, Advanced Encryption Standard, Alpha Press, 2009.

[21] F. Guo, Y. Mu, W. Susilo, D. S. Wong, V. Varadharajan, Cp-abe with constant-size keys for lightweight devices, IEEE Transactions on Information Forensics and Security 9 (5) (2014) 763–771. doi:10.1109/TIFS.2014.2309858.

[22] M. Morales-Sandoval, A. Diaz-Perez, DET-ABE: A Java API for data confidentiality and fine-grained access control from attribute based encryption, in: R. N. Akram, S. Jajodia (Eds.), Information Security Theory and Practice: 9th IFIP WG 11.2 International Conference, WISTP 2015, Heraklion, Crete, Greece, August 24-25, 2015. Proceedings, Springer International Publishing, Cham, 2015, pp. 104–119. doi:10.1007/978-3-319-24018-3_7.
URL http://dx.doi.org/10.1007/978-3-319-24018-3_7

[23] K. Fan, S. Lou, R. Su, H. Li, Y. Yang, Secure and private key management scheme in big data networking, Peer-to-Peer Networking and Applications (2017) 1–8.

[24] A. Rosenthal, P. Mork, M. H. Li, J. Stanford, D. Koester, P. Reynolds, Cloud computing: a new business paradigm for biomedical information sharing, Journal of biomedical informatics 43 (2) (2010) 342–353.

[25] G. Zhao, C. Rong, J. Li, F. Zhang, Y. Tang, Trusted data sharing over untrusted cloud storage providers, in: Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, IEEE, 2010, pp. 97–103.

[26] D.-G. Feng, M. Zhang, Y. Zhang, Z. Xu, Study on cloud computing security, Journal of software 22 (1) (2011) 71–83.

[27] M. Li, S. Yu, Y. Zheng, K. Ren, W. Lou, Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption, IEEE transactions on parallel and distributed systems 24 (1) (2013) 131–143.

[28] C.-K. Chu, S. S. Chow, W.-G. Tzeng, J. Zhou, R. H. Deng, Key-aggregate cryptosystem for scalable data sharing in cloud storage, IEEE transactions on parallel and distributed systems 25 (2) (2014) 468–477.

[29] X. Liu, Y. Zhang, B. Wang, J. Yan, Mona: Secure multi-owner data sharing for dynamic groups in the cloud, IEEE transactions on parallel and distributed systems 24 (6) (2013) 1182–1191.

[30] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: Proceedings of the 2007 IEEE Symposium on Security and Privacy, SP '07, IEEE Computer Society, Washington, DC, USA, 2007, pp. 321–334. doi:10.1109/SP.2007.11.
URL http://dx.doi.org/10.1109/SP.2007.11

[31] P. S. Gardiyawasam Pussewalage, V. A. Oleshchuk, A distributed multi-authority attribute based encryption scheme for secure sharing of personal health records, in: Proceedings of the 22Nd ACM on Symposium on Access Control Models and Technologies, SACMAT '17 Abstracts, ACM, New York, NY, USA, 2017, pp. 255–262. doi:10.1145/3078861.3078880.
URL http://doi.acm.org/10.1145/3078861.3078880

[32] M. I. H. Sukmana, K. A. Torkura, C. Meinel, H. Graupner, Redesign cloudraid for flexible and secure enterprise file sharing over public cloud storage, in: Proceedings of the 10th International Conference on Security of Information and Networks, SIN '17, ACM, New York, NY, USA, 2017, pp. 3–10. doi:10.1145/3136825.3136882.
URL http://doi.acm.org/10.1145/3136825.3136882

[33] S. Zickau, D. Thatmann, A. Butyrtschik, I. Denisow, A. Kï£¡ppe, Applied attribute-based encryption schemes, in: 19th International ICIN Conference - Innovations in Clouds, Internet and Networks, 2016, pp. 88–95.

[34] K. Shahina, D. Lal, A survey on secure data sharing methods in public cloud computing, Proc. Of International Journal of Science, Engineering and Technology Research 5 (01).

[35] M. Sepczuk, Z. Kotulski, A new risk-based authentication management model oriented on user's experience, Computers and Security 73 (2018) 17 – 33.

[36] J. L. Gonzalez-Compean, V. J. Sosa-Sosa, A. Diaz-Perez, J. Carretero, R. Marcelin-Jimenez, Fedids: a federated cloud storage architecture and satellite image delivery service for building dependable geospatial platforms, International Journal of Digital Earth 0 (0) (2017) 1–22.

[37] J. Gonzalez-Compean, V. Sosa-Sosa, A. Diaz-Perez, J. Carretero, J. Yanez-Sierra, Sacbe: A building block approach for constructing efficient and flexible end-to-end cloud storage, Journal of Systems and Software 135

(2018) 143 – 156.

[38] G. A. Vazquez-Martinez, J. Gonzalez-Compean, V. J. Sosa-Sosa, M. Morales-Sandoval, J. C. Perez, Cloudchain: A novel distribution model for digital products based on supply chain principles, International Journal of Information Management 39 (2018) 90 – 103.

[39] K. Gai, M. Qiu, Z. Ming, H. Zhao, L. Qiu, Spoofing-jamming attack strategy using optimal power distributions in wireless smart grid networks, IEEE Transactions on Smart Grid 8 (5) (2017) 2431–2439. doi:10.1109/TSG.2017.2664043.

[40] K. Gai, K. R. Choo, M. Qiu, L. Zhu, Privacy-preserving content-oriented wireless communication in internet-of-things, IEEE Internet of Things Journal 5 (4) (2018) 3059–3067. doi:10.1109/JIOT.2018.2830340.

[41] K. Gai, M. Qiu, Blend arithmetic operations on tensor-based fully homomorphic encryption over real numbers, IEEE Transactions on Industrial Informatics 14 (8) (2018) 3590–3598. doi:10.1109/TII.2017.2780885.

[42] h.-v. a. . The Software Engineering Institute Carnegie Mellon University, U.s. state of cybercrime collection.

[43] P. A. NORBERG, D. R. HORNE, D. A. HORNE, The privacy paradox: Personal information disclosure intentions versus behaviors, Journal of Consumer Affairs 41 (1) (2007) 100–126.

[44] N. L. C. A. R. J.-K. A. M.-W. Bartoletti, D., Predictions 2016: The cloud accelerates, In: Forester Research Technical Report, Forrester Inc.

[45] C. X. T. Zhang, S. Liu, H. Lu, Mining semantic context information for intelligent video surveillance of traffi scenes, IEEE Transantions on Industrial Informatics 9 (1) (2013) 149–160.

[46] J. P. P. B.-W. W. P. Ping, H. Hermjakob, Biomedical informatics on the cloud: A treasure hunt for advancing cardiovascular medicine, Circulation Research 122 (2018) 1290–1301. doi:10.1161/CIRCRESAHA.117.310967.

[47] J. Long, Y. Juntao, A novel clinical decision support algorithm for constructing complete medication histories, Comput. Methods Prog. Biomed. 145 (C) (2017) 127–133. doi:10.1016/j.cmpb.2017.04.004.

[48] S. Simske, H. Balinsky, Apex: Automated policy enforcement exchange, in: Proceedings of the 10th ACM Symposium on Document Engineering, DocEng '10, ACM, New York, NY, USA, 2010, pp. 139–142. doi:10.1145/1860559.1860587.

[49] J. M. A. Calero, G. M. Perez, A. F. G. Skarmeta, Towards an authorisation model for distributed systems based on the semantic web, IET Information Security 4 (4) (2010) 411–421. doi:10.1049/iet-ifs.2009.0260.

[50] E. M. A. Shatnawi, T. Cheng, Maintaining integrity and non-repudiation in secure offline documents, in: Proceedings of the 2017 ACM Symposium on Document Engineering, DocEng '17, ACM, New York, NY, USA, 2017, pp. 59–62. doi:10.1145/3103010.3121038.

[51] P. Y. R. Wu, L. Huang, H. Zhou, Edaws: A distributed framework with efficient data analytics workspace towards discriminative services for critical infrastructures, Future Generation Computer Systems 81 (2018) 78 – 93. doi:https://doi.org/10.1016/j.future.2017.11.009.

[52] K. Xue, P. Hong, A dynamic secure group sharing framework in public cloud computing, IEEE Transactions on Cloud Computing 2 (4) (2014) 459–470.

[53] J. Shen, D. Liu, J. Shen, Q. Liu, X. Sun, A secure cloud-assisted urban data sharing framework for ubiquitous-cities, Pervasive and mobile Computing 41 (2017) 219–230.

[54] A. Gandomi, M.Haider, Beyond the hype: Big data concepts, methods, and analytics, International Journal of Information Management 35 (2015) 137–144.

[55] IDG, 2016 data and analytics research.

[56] Y. B. B. C.-M. L. G. Quellec, K. CharriEre, Deep image mining for diabetic retinopathy screening, Medical Image Analysis 39 (2017) 178–193.

[57] D. P. Y. Guo, X. Jia, Effective sequential classifier training for svm-based multitemporal remote sensing image classification, IEEE Transactions on Image Processing 27 (2018) 3036–3048.

[58] W. Rawat, Z. Wang, Deep convolutional neural networks for image classification: A comprehensive review, Neural Computation 29 (9) (2017) 2352–2449.

[59] S. P. K. et al., Emonets: Multimodal deep learning approaches for emotion recognition in video, Journal on Multimodal User Interfaces 10 (2) (2016) 99–111.

[60] R. Sharan, T. Moir, An overview of applications and advancements in automatic sound recognition, Neurocomputing 200 (2016) 22–34.

[61] K. N. H. O.-T. O. K. Noda, Y. Yamaguchi, Audio-visual speech recognition using deep learning, Applied Intelligence 42 (4) (2015) 722–737.

[62] Z. S. Harris, Distributional structure. in: The philosophy of linguistics, Word 10 (2-3) (1954) 146–162.

[63] J. Weeds, D. Weir, Co-occurrence retrieval: A flexible framework for lexical distributional similarity, Computational Linguistics 31 (4) (2005) 439–475.

[64] M. E. Newman, Detecting community structure in networks, The European Physical Journal B-Condensed Matter and Complex Systems 38 (2) (2004) 321–330.

[65] B. Rosenberg, Handbook of Financial Cryptography and Security, 1st Edition, Chapman & Hall/CRC, 2010.

[66] J. Gonzalez, J. C. Perez, V. J. Sosa-Sosa, L. M. Sanchez, B. Bergua, Skycds: A resilient content delivery service based on diversified cloud storage, Simulation Modelling Practice and Theory 54 (Complete) (2015) 64–85.

[67] O. T.-H. Victor J. Sosa-Sosa, Miguel Morales-Sandoval, J. L. Gonzalez-Compean, Protecting data in the cloud: an assessment of practical digital envelopes from attribute based encryption, 6th International Conference on Data Science, Technology and Applications.

Authors

J. L. Gonzalez-Compean received his Ph.D. in Computer architecture from UPC Universitat Politècnica de Catalunya, Barcelona (2009). He was a visiting professor at Universidad Carlos III de Madrid, Spain and researcher at Cinvestav, Tamaulipas, Mexico. His research lines are Cloud-Based Storage systems, linguistic archival systems, federated storage networks. His expertise areas are the design of fault-tolerant, adaptability and availability strategies as well as task scheduling and storage virtualization.

Victor J. Sosa-Sosa is a full-time research-professor at CINVESTAV Tamaulipas - IPN. He has a Ph.D. in Computer Science from Technical University of Catalonia (UPC-Barcelona). He was professor and researcher in the Distributed Systems Group at CENIDET-Mexico until 2006. Participant and responsible of research and development projects funded by the National Council for Science and Technology of Mexico (CONACYT) and private companies. His research interest and specialization areas are related to Distributed Information Systems and Databases. Recently, his work is more focused on distributed information at Web scale, especially on topics related to efficient information search, storage and analysis, integration of Web query interfaces (DeepWeb) and knowledge harvesting.

Oscar Telles received master degree by Information Technology Laboratory, Center of Research and Advanced Studies of the National Polytechnic Institute, (CINVESTAV), Ciudad Victoria, Mexico. His research areas of interest are data analysis and cloud computing.

Ivan Lopez-Arevalo is an associate professor at the Information Technology Laboratory from Cinvestav (Center for Research and Advanced Studies of the National Polytechnic Institute) at Ciudad Victoria, Tamaulipas, Mexico. His research areas of interest are data mining: knowledge representation, knowledge management and ontologies. Expertise in semantic Search on the Web and Text Mining on digital libraries, Decision Support Systems for Clinical Practices Guidelines and Machine Learning for data analysis.

Miguel Morales-Sandoval received PhD degree in Computer Science in 2008 from the National Institute for Astrophysics, Optics and Electronics (INAOE), in the Computer Science Department. Postdoctoral fellow inthe Centerfor Research andAdvanced Studies of the National Polytechnic Institute (Cinvestav) campus Tamaulipas from 2012 to 2014. He was a visiting professor from 2014 to 2016 at Cinvestav and now he is a full time professor-researcher (Cinvestav-3B level) in Cinvestav Tamaulipas. His research areas are cryptographic algorithms and protocols for embedded systems, data security in constrained devices and in the cloud, embedded system design with HDLs and FPGAs, reconfigurable computing as well as software and hardware engineering.

Jesus Carretero Perez is a senior full time professor-researcher at Carlos III University, Computer Science and Engineering department, Madrid, and his major research lines are high-performance computing and cloud computing, parallel and distributed systems, computational linguistics and real-time systems. He leader of research group ARCOS at Carlos III University and he is also the director of the Master in Administration and Management of Information systems and has published 17 books, has been involved in 52 research projects and he has written 200 journal and congress articles.

Authors

J. L. Gonzalez-Compean

Victor J. Sosa-Sosa

Ivan Lopez-Arevalo

Miguel Morales-Sandoval

Jesus Carretero Perez

Highlights

- SecFilter implements security information policy to mitigate privacy paradox effects
- Data mining, risk assessment and encryption schemes are integrated into SecFilter
- SecFilter automatically analyzes and secures files before sending them to the cloud
- Implementation of SecFilter is based on virtual containers to produce task parallelism
- A prototype of SecFilter was evaluated through a fine-tuning and case study.