UNIVERSITY OF AGDER

Privacy-preserving document similarity detection

Ksenia Khelik

Supervisor
Vladimir Oleshchuk

This Master's Thesis is carried out as a part of the education at the University of Agder and is therefore approved as a part of this education

University of Agder, 2011
Faculty of Engineering and Science
Department of ICT

Abstract

The document similarity detection is an important technique used in many applications. The existence of the tool that guarantees the privacy protection of the documents during the comparison will expand the area where this technique can be applied. The goal of this project is to develop a method for privacy-preserving document similarity detection capable to identify either semantically or syntactically similar documents. As the result two methods were designed, implemented, and evaluated. In the first method privacy-preserving data comparison protocol was applied for secure comparison. This original protocol was created as a part of this thesis. In the second method modified private-matching scheme was used. In both methods the Natural Language processing techniques were utilized to capture the semantic relations between documents. During the testing phase the first method was found to be too slow for the practical application. The second method, on the contrary, was rather fast and effective. It can be used for creation of the tool for detecting syntactical and semantic similarity in a privacy-preserving way.

Thesis definition

Privacy-preserving document similarity detection

Group 13: Ksenia Khelik

Detection of document similarity is an important problem with many applications in the areas of for instance plagiarism detection, copyright protection, file management, document searching etc. There can be distinguished two types of document similarity: syntactic and semantic one. The documents are syntactically similar if they are written with the same words. The documents are semantically similar if they contain the same information. Many tools have been developed for documents comparisons based both on semantic and syntactic analysis.

In this project we consider the problem of privacy-preserving similarity detection of documents. Such solution is needed when users want to compare documents without disclosing them to each other. The privacy-preserving similarity detection implies existence of the secure protocol that protects document contexts from disclosing to the other side during the comparison. The goal of this master project is to design and implement such solution in order to create a tool that can be used for privacy-preserving document comparison.

Contents

List of figures

List of tables

1 Introduction

The document similarity detection is used in many different applications, such as plagiarism detection, copyright protection and file management. A lot of methods are developed and many tools, i.e. [23], [24], [25], were created in order to effectively compare the documents, but only few of these methods ([6], [10], [14]) and none of tools can be applied to confidential document similarity detection. The handling of the documents of such type implies the comparison process is organized in such way that it guarantees the privacy protection of document contents. The existing methods, satisfying this condition, reveal only the syntactically similar confidential documents. In this project it was attempted to develop a method which along with a syntactic similarity can also detect a semantic one in the privacy-preserving way.

1.1 Background

The document similarity detection is an important technique used in many applications. It can be applied to efficient data management, to detection of copyright violations and plagiarism. But only few of the existing methods and none of tools provide privacy protection of the documents they compare. Development of the method for privacy-preserving documents comparison enables to expand the area where document similarity detection can be applied. Some examples of privacy-preserving similar document detection usage are given below.

At first it can be used in universities to prevent plagiarism in project reports. Sometimes project can be given by the company. Working on such project, a student can get an access to some company's confidential documents and use their parts in his or her report without references. The creation of a common database, consisting of companies' documents, related to the project topics, can help to deal with this problem. The presence of a secure protocol for data transfer and comparison will provide documents confidentiality. At second, this technique can be applied to reveal duplicated conference registrations or paper submissions. Nowadays, there is no effective procedure to detect the paper that is submitted to obtain the permission for the author to participate in different conferences or to be published in different journals. Privacy-preserving document similarity detection enables to unite conference committees or journal redactors to find dishonest applicants. Thirdly, this tool can help the customers to decide whether or not they need to buy the chosen articles or books. In many online book shops there is no preview option available, so sometimes it is impossible to check if this article or book is the same one that the customer already has or how many changes the authors have made in their new editions as compared to the previous ones. Opportunities, the privacy-preserving document similarity detection provides, allows the customer to resolve doubts before buying without infringing the copyrights.

It can be distinguished 4 kinds of document similarity: semantic, syntactic, lexical and structural one. In this project only semantic and syntactic similarities are taken into consideration. Two documents are syntactically similar if they are written with the same words placed in the same order. The documents are semantically similar if they contain the same information possibly written with different words. Unfortunately, it is not always enough to check only whether the documents are

syntactically similar. Very often different changes are applied to the document to hide the fact that this document is a copy of another one. The most widespread methods are: changing the order of words, sentences, paragraphs, using synonyms, paraphrasing the sentences, adding and/or deleting some parts. So in order to compare the documents correctly, it is also necessary to check the presence of semantic document similarity. The existence of the method, that detects both semantic and syntactic similarity and guarantees the privacy protection for the confidential documents, will improve the quality of document comparison and expand the area where document similarity detection can be applied.

1.2 Problem statement

Let Alice and Bob represent two parties each of whom has a collection of documents. The document is considered as a text, paragraph, sentence or just several distinct words. Both Alice's and Bob's documents are assumed to be confidential. The goal of this project is to develop a method that enables to detect whether or not Bob's collection contains a document, similar to the some document from Alice's collection without disclosing Bob's data to Alice and vice-versa. This method should indentify either semantically or syntactically similar documents.

The aim implies the creation of a secure protocol for privacy-preserving data comparison. This protocol should be design in such manner that it doesn't require the presence of the trusted third party.

1.3 Literature review

Several approaches were proposed in order to detect the syntactically similar documents. One of the widely used methods is to introduce vector space model [1], [2], [3]. All words, met in both documents, form the global vector space. Then the individual vector for each document is created. This vector consists of elements representing how much times the words from global space appeared in the text. More common words the documents share, closer the vectors are. To find the distance between the vectors the cosine theorem [4] can be used. If the distance is greater than some pre-defined threshold, then the documents are considered to be similar.

Some attempts were made to use vector space model for privacy-preserving document comparison. In [5] the authors proposed a method for similarity-based text retrieval that safeguard the content of the user queries and the retrieved documents. To provide the privacy the trusted third party was involved. In [6], [7] two different ways for privacy-preserving distance finding were described. The first one is based on random matrix-based privacy-preserving dot product protocol [8]. The second way is to employ the properties of homomorphic encryption. Both of these methods guarantee the privacy of the individual vectors and, consequently, the document contents.

The comparison of the fingerprints is an alternative way to find a syntactical document similarity. The idea to map a text into a bit-string, called fingerprint, was introduced in [9] by M.O. Rubin. Later in [10] A. Z. Broder showed how Rubin's fingerprints can be applied for similar document detection. Since then, some work has been done to optimize this approach. In [11] the attack-resistant

method for identifying plagiarized documents was presented. The security was provided by unpredictable fingerprint counting process. The algorithms, described in [12], [13], were able to detect near-duplicated documents. As fingerprints comparison doesn't reveal the document contents to another party, it can be used for privacy-preserving similarity detection.

One more approach, called private matching scheme, was described in [14]. It was created to find the intersection of private datasets of two parties. The method is based on the homomorphic encryption properties that are used for operation performed on polynomial whose roots are the dataset elements of one of the party. Unfortunately, this scheme doesn't provide confidentiality to all elements in datasets, but can be easy modified to be applied to privacy-preserving document similarity detection.

As for revealing semantic similarity in the documents, the usage of the natural language processing techniques is one of the effective ways to do it. In [15] it was proposed to parse the document to extract the distinct words, called tokens. The parsing process includes the removal of high frequency words and the stemming of the remaining words. The obtained sets of tokens are used for documents comparison. In [16] it was experimentally shown that it is enough to compare only nouns, adjectives and verbs to detect the semantic similarity of the documents. The authors of [17], [18] suggested to use the ontology, as it enables to reflect semantic relationship between documents and make comparison process more effective. The overview of all natural language processing techniques that can be used for semantic similarity detection is given in [19].

1.4 Problem solution

The analysis of the existing methods for different document similarity types detection showed there is no such method that reveals both semantically and syntactically similar documents in privacy-preserving way. So it was decided to combine some found approaches to be able to do it. As the result two methods were developed. Both of these methods have the following structure. At first for all documents of each party the areas, the documents are related to, are found employing ontology. Then every document is processed, using Natural Languages Processing techniques, and transformed into a set of distinct meaningful words, called footprint. After that footprint and areas of each document of one party are compared respectively with a footprint and areas of every document of another party in privacy-preserving way. In the first method privacy-preserving data comparison protocol, created as a part of this thesis, was employ for secure comparison. In the second method private matching scheme [14] was applied after some modifications for this purpose. For methods implementation Python programming language was used.

2 Theoretical background

As it was mentioned in paragraph 1.4, two methods for privacy-preserving document similarity detection were developed. To create them, several approaches were used. They are Natural Language Processing techniques [19], private matching scheme [14], and privacy-preserving data comparison protocol. The description of these approaches is given in this chapter.

2.1 Natural Language Processing

Natural Language Processing (NLP) is a theoretically motivated range of computational techniques for analyzing and representing texts in natural human-languages [20]. All NLP systems aim to achieve a human-like performance during language processing. For this purpose they use different levels of linguistic analysis utilized by humans to produce or comprehend language. The following levels of linguistic analysis can be distinguished [20]: phonology, morphology, lexical, syntactic, semantic, discourse, and pragmatic one.

Phonology level deals with the interpretation of speech sounds within and across words. The NLP systems analyze the sound waves and encoded them into a digitized signal for further processing. Morphology level analyzes the form and structure of words. All words are composed of the smallest units of meaning, called morphemes. Since the meaning of each morpheme remains the same across words, the NPL systems exploit this property to gain the meaning of the unknown words. At the lexical level the sentences are divided into separate words. The NLP system assigns the part-of-speech tag to each word according to the context in which this word occurs and then determines its possible meanings. Syntactic level focuses on analyzing the words in a sentence in order to uncover the structural dependency between the words. The semantic level goal is to determine the meaning of the sentences based on the words meanings. This stage includes the semantic disambiguation of polysemous words. Discourse and pragmatic levels are the highest levels of processing. Discourse is focusing on finding of the connections between sentences. Pragmatic level is concentrated on the context over and above the text contents for better text understanding [20].

The NLP techniques are used in many applications, such as information retrieval, text mining, language understanding, and text classification. The description of the often-used techniques is given below.

2.1.1 Natural Language Processing techniques

Stop list

Stop list contains high frequency words, such as ’of’, ‘a’, or ‘is’. Usually these words are ignored to improve the performance. But in some applications, i.e. text mining, they can help to determine a part of speech or to clarify the semantics of the text segment [19].

Stemming

Stemming is the process of reducing the word to its root. For example, ‘derives’ and ‘derivation’ can be replaced by ‘deriv’. It leads to the reducing of the number of meaningful words, but can also affect the semantics.

Lemmatization

Lemmatization is the process of reducing the word to its canonical form. For example, ‘is’, ’are’, and ‘been’ will be replaced by ‘be’. As well as stemming, it leads to reducing the number of meaningful words, but make it in more proper way, using vocabulary and morphology analysis of the words [21].

Noisy data

Noisy data refers to the words with spelling mistakes, acronyms and abbreviations. It can be very useful to correct all mistakes and to replace all acronyms and abbreviations at the beginning of analysis.

Word sense disambiguation

The word sense disambiguation problem is about finding out the most probable meaning of a polysemous word [19]. Several approaches can be used to deal with this problem: dictionary and knowledge based, supervised, and unsupervised methods. Dictionary and knowledge based methods suppose that the sense of the word can be extracted from definitions of the words, so they use dictionaries, thesauri and other lexical knowledge bases. Supervised methods mainly adopt context to disambiguate words. These methods include training and testing phases. In the training phase, a sense-annotated training corpus is required, from which syntactic and semantic features are extracted to create a classifier using machine learning techniques [22]. When lexical knowledge bases or training data are unavailable, the unsupervised methods are used. These methods acquire contextual information directly from unannotated raw text, and senses can be induced from text using some similarity measure [22].

Part of speech tagging

Part of speech (POS) tagging is a process of assigning a part of speech to each word in a sentence. It helps to determine the right word sense. There following basic POS tags are used by all taggers: verb, noun, pronoun, adjective, adverb, preposition, conjunction, and interjection with some taggers adding the article [16]. Many tagging systems extend these basic tags to describe additional grammatical features, such as singular/plural, number, tense, gender, and even punctuation [16].

Ontology

Ontology is a knowledge structure that specified terms, their properties and relations among them to enable knowledge extraction from the text [18]. It consists of concept, concept-relations, axiom and instances [19].The selection of concepts depends on the task and the domain information that need to be captured. Thus before constructing the ontology, it is important to know what it will be used for. The ontology reflects the structure of the certain domain and clarifies the meaning of thespecial terms appeared in the text. Thus it can be used to provide expert, background knowledge about the domain and constrain the possible senses of the terms. The application of

ontology to the document comparison can reveal the semantic similarity that hasn't been noticed before [18].

Tokenization

Tokenization is the process of breaking up the sequence of characters in a text by locating the word boundaries, the points where one word ends and another begins [26]. The derived words are called tokens. Tokenization is rather easy task for the languages, like English, Russian, or German, in which words are separated by space character. But it can be complicated for Chinese and some other languages, because a word may be represented as a single character or series of several characters, and there may be no space between words [26].

2.2 Private matching scheme

Private matching (PM) scheme was proposed in [14] by B. Pinkas, M.J. Freedman and K. Nissim, to find the intersection of private datasets elements taken from the same domain. It is a two-party protocol based on the use of encryption function Enc, satisfying following properties:

(1) Encryption function is a homomorphic function:
Given two ciphertexts $Enc(m_1)$ and $Enc(m_2)$, then $Enc(m_1 + m_2) = Enc(m_1) \oplus Enc(m_2)$, for some operation $\oplus$.

(2) It allows multiplication by a constant:
Given ciphertext $Enc(m)$ and some constant c, then it's possible to compute $Enc(c \cdot m)$, without decrypting ciphertext

As an example of the encryption function, satisfying these properties, Paillier's cryptosystem [27] can be taken. It is described in paragraph 2.2.2.

2.2.1 Private matching scheme description

Let Alice and Bob represent two parties. Alice's input is a set $X = \{x_1, \dots, x_N\}$ from some domain, Bob's input is a set $Y = \{y_1, \dots, y_K\}$ from the same domain. According to [14] protocol has the following basic structure. Alice defines a polynomial P, whose roots are her inputs:

$$P(t) = (x_1 - t)(x_2 - t) \dots (x_N - t) = \sum_{u=0}^{N} a_u t^u$$

She sends to Bob encrypted coefficients of this polynomial. Bob employs the homomorphic properties of the encryption system to evaluate the polynomial at each of his inputs. Then he multiplies each result by a random number r, adds to it an encryption of the value of his input, to get $Enc(r_1 \cdot P(y_1) + y_1), \dots, Enc(r_K \cdot P(y_K) + y_K)$ and sends them to Alice. She decrypts received values and compares obtained values with her inputs. The result is a number of common elements and their values. Other decrypted values, not equal to some Alice's element values, are just random numbers, and don't reveal the corresponding Bob's element values. The scheme is

organized in such way that Bob can't know the Alice's element values. The detailed protocol description is given Figure 1.

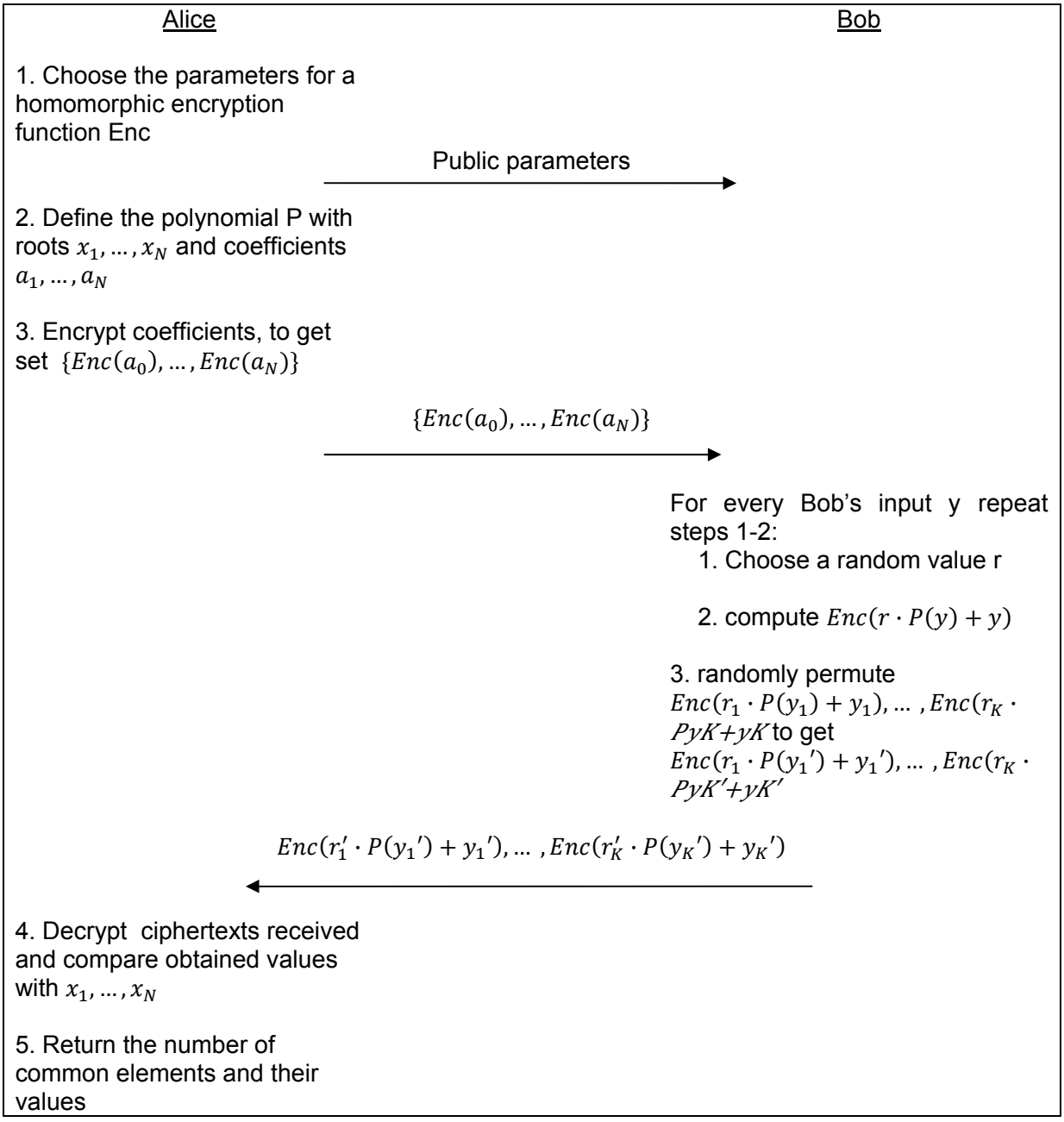


Figure 1: Private matching scheme outline

2.2.2 Paillier's cryptosystem

Paillier's cryptosystem was first proposed in [27]. This is a public-key encryption, based on usual modular arithmetic.

Key generation

1. Choose two large prime numbers p and q

2. Set $n = p \cdot q$ and $\lambda = lcm(p-1, q-1)$
3. Randomly select $g \in \mathbb{Z}_{n^2}^*$: $\gcd(\frac{g^\lambda mod n^2 - 1}{n}, n) = 1$
4. Consider n, g as public parameters and p, q as private ones.

Encryption

1. Given message $m \in \mathbb{Z}_n^*$
2. Select a random $r < n$
3. Compute ciphertext $c = g^m \cdot r^n \ mod \ n^2$

Decryption

1. Given ciphertext $c \in \mathbb{Z}_{n^2}^*$
2. Compute plaintext $m = L\left(c^\lambda \ mod \ n^2\right) \cdot \left(L\left(g^\lambda \ mod \ n^2\right)\right)^{-1} mod \ n$, where $L(u) = \frac{u-1}{n}$

Properties

Paillier's encryption function satisfies the following properties:

1. It is a homomorphic function.

Proof:

$\forall \ m_1, m_2 \in \ Z_n^*$:

$$Enc(m_1 + m_2) = g^{m_1+m_2} \cdot r^n \ mod \ n^2 = g^{m_1} \cdot g^{m_2} \cdot r^n \ mod \ n^2 =$$
$$= (g^{m_1} \cdot r^n \ mod \ n^2) \cdot (g^{m_2} \cdot r^n \ mod \ n^2) mod \ n^2 = Enc(m_1) \cdot Enc(m_2) mod \ n^2$$

□

2. It allows multiplication by a constant

Proof: $\forall \ m \in Z_n^*, k \in \mathbb{N}$ "

$$Enc(k \cdot m) = g^{k \cdot m} \cdot r^n \ mod \ n^2 = (g^m)^k \cdot r^n \ mod \ n^2 = (g^m \cdot r^n \ mod \ n^2)^k \ mod \ n^2 =$$
$$= (Enc(m))^k mod n^2$$

□

Thus Paillier's encryption function satisfies the properties listed in paragraph 2.2, so it can be used in private matching scheme described in paragraph Private matching scheme description 2.2.1

2.3 Privacy-Preserving Data Comparison Protocol

This original privacy-preserving data comparison protocol was created as an alternative way to perform documents comparison in privacy-preserving way. This protocol is based on the use of commutative encryption, that is:

If Enc_1 and Enc_2 are two commutative encryption functions and m is a plaintext, then ciphertexts $Enc_1(Enc_2(m))$ and $Enc_2(Enc_1(m))$ are equal.

As the example of commutative encryption, The Massey-Omura cryptosystem [28] can be used. It is described in paragraph 2.3.2.

2.3.1 Privacy-preserving Data Comparison Protocol Description

Let Alice and Bob represent two parties. Alice has a set of elements $X = \{x_1, \dots, x_N\}$. Bob has a set of elements $Y = \{y_1, \dots, y_N\}$. Protocol is organized as follows.

Alice and Bob define the commutative encryption functions Enc_1 and Enc_2, respectively. Both of them encrypt their elements with the corresponding encryption function and send the obtained sets to each other. Having received the encrypted Alice's elements, Bob encrypts them, using Enc_2, permutes and sends to Alice. Meanwhile, Alice encrypts Bob's encrypted elements, using Enc_1. Then she receives back her elements, modified by Bob. At the moment Alice has two sets: $Enc_2Enc_1(X')$ and $Enc_1Enc_2(Y)$. Because of commutative encryption property, the ciphertexts of the elements, that contains in X and Y set intersection, are identical. So the number of common elements in X, Y sets are the same as the number of common elements in $Enc_2Enc_1(X')$ and $Enc_1Enc_2(Y)$ sets. All Alice needs to get a result is to compare these new sets. Since Bob has shuffled Alice's encrypted elements, Alice can't determine which of her elements are identical to Bob's ones. The detailed protocol description is given in Figure 2.

Alice		Bob
1. Choose the parameters for a homomorphic encryption function Enc_1		1. Choose the parameters for a homomorphic encryption function Enc_2
2. Encrypt Alice's elements with Enc_1 to get set $E_1(X)$		2. Encrypt Bob's elements with Enc_2 to get $E_2(Y)$
	$E_1(X)$ →	
	← $E_2(Y)$	
3. Encrypt $E_2(Y)$ with Enc_1 to get set $E_1E_2(Y)$		3. Encrypt $E_1(X)$ with Enc_2 to get set $E_2E_1(X)$
		4. Randomly permute $E_2E_1(X)$ to get set $E_2E_1(X')$
	← $E_2E_1(X')$	
4. compare $E_1E_2(Y)$ and $E_2E_1(X')$ 5.return the number of common elements		

Figure 2: Privacy-preserving data comparison protocol outline

2.3.2 Massey-Omura cryptosystem

The Massey-Omura cryptosystem was described in [28]. It was created for secure data transfer between 2 parties. Let Alice and Bob represent two parties. Alice needs to send message M to Bob. The message transfer scheme is described below.

Key generation:

1. Alice choose a large prime number q
2. q is considered as a public key
3. Alice selects a random positive integer e_1, such that $e_1 < q$ and $gcd(e_1, q-1) = 1$, and computes $d_1 = e_1^{-1} mod\ q - 1$
4. Bob selects a random positive integer e_2, such that $e_2 < q$ and $gcd(e_2, q-1) = 1$, and computes $d_2 = e_2^{-1} mod\ q - 1$
5. e_1, d_1 are Alice's private keys, e_2, d_2 are Bob's private keys.

Encryption:

1. Given message $m \in \mathbb{Z}_q^*$
2. Let e be a private key
3. Compute ciphertext $c = M^e\ mod\ q$

Decryption:

1. Given ciphertext $c \epsilon \mathbb{Z}_q^*$
2. Let d be a private key, such that $d = e^{-1}\ mod\ q - 1$
3. Compute message $m = c^d\ mod\ q$

Message transfer:

The Massey-Omura cryptosystem requires three messages to be sent to achieve a secure transmission. This process is shown in Figure 3.

Alice		Bob
Alice needs to send a message M		
	1. $A = M^{e_1} mod\ q$ →	
	2. B= $A^{e_2} mod\ q$ ←	
	3. C= $B^{d_1} mod\ q$ →	
		4. Bob calculates $D = C^{d_2} mod\ q$ and gets the message M, that is equal to D

Figure 3: Massey-Omura message transfer protocol

Properties

Massey-Omura cryptosystem satisfies the property of commutative encryption.

Proof:

Let q be a public key. Enc_1 is an encryption function with e_1 private key, and Enc_2 is an encryption function with e_2 private key. Given message $m \in \mathbb{Z}_q^*$. Then

$Enc_1\big(Enc_2(M)\big) = Enc_1(M^{e_2} \bmod q) = (M^{e_2})^{e_1} \bmod q = M^{e_1 \cdot e_2} \bmod q =$

$= (M^{e_1})^{e_2} \bmod q = Enc_2(M^{e_1} \bmod q) = Enc_2\big(Enc_1(M)\big)$

□

Thus, Massey-Omura encryption functions satisfy the property mentioned in paragraph 2.3. So they can be used in protocol described in paragraph 2.3.1.

3 Solution

Within the bounds of IKT590 project, two methods for privacy-preserving documents similarity detection were designed, implemented and tested. The description of these steps is given in this chapter.

3.1 Requirements

As it was said in paragraph 1.2, the goal of this project is to develop a method for privacy-preserving document similarity detection. This method should satisfy the following requirements:

General requirements:

1. It should take as little time as possible.
2. The number of data transfers should be minimized.

Functional requirements:

1. It should find semantically similar documents.
2. It should find syntactically similar documents.
3. It should detect a document that is a part of another one.
4. It should detect documents that contain common part.
5. It should be possible to compare several documents at once.

Security requirements:

1. No third party should be involved.
2. It is allowed to know the number of words, participating in comparison.
3. Every party have a right to know the number of words, containing in both compared documents.
4. There should be no way to know the words, containing in both compared documents.
5. There should be no way for each party to know the words of another party.

3.2 Input Data

For each party input data are one or several documents in .txt format. The documents should be written in English language. The tables and drawing are not processed. The documents, containing none or one word, don't participate in the comparison process.

3.3 Output Data

The output data shows the type of similarity relationship between each document of one party and each document of another party. Let Doc_1 and Doc_2 are two documents, have been compared. The similarity relationship types between them can be following:

1. The documents are syntactically similar (duplicates)
2. The documents are almost syntactically similar (near duplicates)
3. The documents are semantically similar

4. The documents share common part
5. Doc_1 is a semantically similar to some part of Doc_2
6. Doc_2 is a semantically similar to some part of Doc_1
7. Documents are different

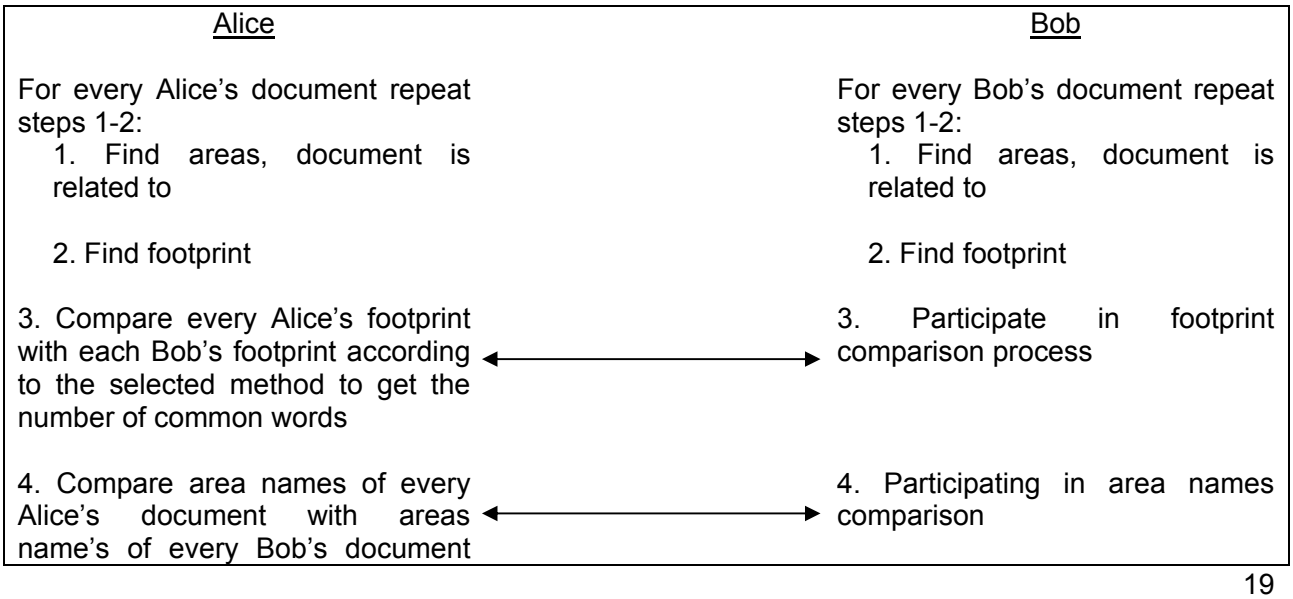
3.4 Design Specification

According to the requirements, given in paragraph 3.1, two methods for privacy-preserving document similarity detection were designed and implemented. In this chapter the detailed overview of these methods can be found.

3.4.1 Methods overview

Let Alice and Bob represent two parties. Alice has a set of documents Doc_A, Bob has a set of documents Doc_B. They want to know the types of similarity relationship between their documents. To find the answer, they use one of two developed methods: Method 1 or Method 2. These methods have similar basic structure.

At first Alice and Bob prepare their documents and determine the areas, the documents are related to, using ontology. Then each party process every document, applying Natural Language Processing techniques described in paragraph 2.1.1, and transforms it to the set of distinct meaningful words, called footprint. According to the method parties have chosen, Alice compares her every footprint with each Bob's one in privacy-preserving way and gets the number of common words in each footprint pair. Also she finds the semantic relationship between her and Bob's documents performing the secure area names comparison. To guarantee the privacy of footprints and area names during comparison process, Method 1 employs the privacy-preserving data comparison protocol, while Method 2 uses the modified private-matching scheme. According to the numbers of common words and semantic relationship, Alice finds the types of similarity relationship between her and Bob's documents. The generalized description of privacy-preserving document similarity detection process is shown in Figure 4.



according to selected method to get the semantic relationship between documents 5. Find result according to the number of common words and semantic relationship	

Figure 4: Privacy-preserving document similarity detection process outline

3.4.2 Areas finding

To find the areas, document is related to, ontology is used. It is one of the Natural Language Processing technique described in paragraph 2.1.1. Ontology consists of areas that, in turn, can contain some other areas. Every area is described by the set of terms in an unique way. Each term can be a phrase or a single word. It is allowed to include the same term in several sets. The ontology structure is presented in Figure 5.

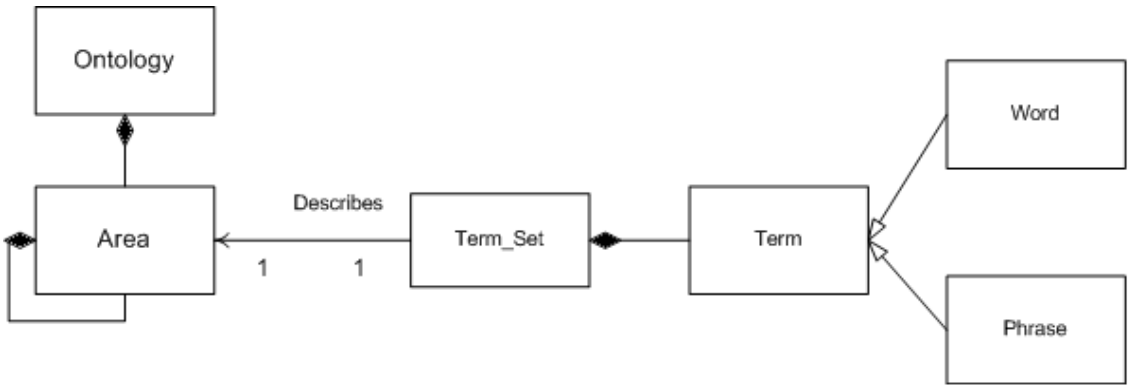


Figure 5: Ontology structure

The areas are determined according to the ontology terms the document contains. The finding process consists of following steps:

For every ontology area repeat steps 1-2:

1. Search for all terms this area is described by.
2. If the number of found terms is greater than some predefined value, add the name of this area to the list of area names

3. Return the list of area names

The threshold was introduced to avoid the wrong area detection in the following cases:

1. There are homonymous words to the ontology terms in the document
2. A few terms from some area are mentioned in the document, but the document actually isn't related to this area.

3.4.3 Footprint finding

Footprint finding is a second step in a privacy-preserving document similarity detection process. It is based on the application of several Natural language processing techniques described in paragraph 2.1.1. This step is performed to transform a document into a set of distinct meaningful words, called footprints. Meaningful words are the words reflecting the semantics of the document content. They are nouns, adjectives and verbs. In [16] it was experimentally shown by the example of English language that it is enough to compare only the words, marked with these parts of speech, in order to detect the semantic similarity of the documents. The removal of other parts of speech allows to increase the algorithm performance during the footprint comparison step. But there is one disadvantage of this approach. The users can't be sure whether the documents are 100% identical in case of document footprints equality. If it is important for the users, the additional word-by-word documents comparison can be performed.

The following Natural Language Processing techniques are used to find document footprint: tokenization, part-of-speech tagging, stemming. All of them, except tokenization, require the presence of the knowledge base.

The footprint finding process starts with the knowledge base preparation (it will be explained in 3.5.1.1). Then all letters are converted to lower-case and the document is divided into separate words, called token. After that every token is marked with appropriate part-of-speech tag and the ones with noun, adjective and verbs tags are selected. The chosen tokens are stemmed. Then repetitions in list of stemmed tokens are deleted and footprint is ready. The overview of the footprint finding process is shown in Figure 6.

1. Prepare knowledge base
2. Convert all letters to lower-case
3. Perform document tokenization
4. Perform part-of-speech tagging
5. Choose words with nouns, verbs and adjectives tags.
6. Perform stemming
7. Delete repetitions

Figure 6: Footprint finding process

3.4.4 Footprint comparison

During the footprint comparison step every footprint of one party is compared with each footprint of another party and number of common words between them is found. The comparison is performed by one of the parties. Let it be Alice. According to the chosen method for privacy-preserving document similarity detection, she can compare the footprints in one of two ways: using privacy-preserving data comparison protocol or applying modified private-matching scheme. The description of Method 1 and Method 2 footprint comparison steps are given below.

3.4.4.1 Method 1 footprint comparison algorithm

Method 1 footprint comparison algorithm is based on privacy-preserving data interchanging protocol described in paragraph 2.3 and has the following structure.

Let Alice has a set of footprints $Footpr_A = \{footpr_{A1}, \dots, footpr_{AN}\}$. Bob has a set of footprints $Footpr_B = \{footpr_{B1}, \dots, footpr_{BK}\}$. The algorithm starts from the footprints preparation for comparison process. Alice divides each footprint into 2 subsets. The first one consists of the words, containing in all her footprints. The second one consists of the rest of the words, not containing in the first subset. Alice transforms every word in the subsets into numerical representation. As the result she gets subsets Com_A, $Diff_{A1}$,…, $Diff_{AN}$. The same actions are performed by Bob and he obtains the subsets Com_B, $Diff_{B1}$,…, $Diff_{BK}$. The footprints are divided into 2 subsets to avoid the encryption and transfer of the words that are in all footprints of one party several times. In some cases, e.g. when all documents are related to the same area, the saving of time can be significant.

Then Alice and Bob define homomorphic encryption functions Enc_1 and Enc_2, respectively. They encrypte their subsets with corresponding encryption functions and send them to each other. Having received the encrypted Alice's subsets, Bob encrypts them with Enc_2, permutes elements in every subset and sends to Alice. Meanwhile, Alice encrypts Bob's encrypted subsets, using Enc_1. Then she receives back her subsets, modified by Bob. At the moment Alice has following sets: $Enc_2Enc_1(Com'_A)$, $Enc_2Enc_1(Diff'_{A1}), \dots, Enc_2Enc_1(Diff'_{AN})$, $Enc_1Enc_2(Com_B)$, $Enc_1Enc_2(Diff_{B1}), \dots$, $Enc_1Enc_2(Diff_{BK})$. She unions $Enc_2Enc_1(Com'_A)$ with $Enc_2Enc_1(Diff'_{A1}), \dots, Enc_2Enc_1(Diff'_{AN})$ to get her encrypted footprints $Enc_2Enc_1(footpr'_{A1}), \dots, Enc_2Enc_1(footpr'_{AN})$ and unions $Enc_1Enc_2(Com_B)$ with $Enc_1Enc_2(Diff_{B1}), \dots, Enc_1Enc_2(Diff_{BK})$ to get Bob's encrypted footprints $Enc_1Enc_2(footpr_{B1}), \dots, Enc_1Enc_2(footpr_{BK})$.Then she compares each of her encrypted footprints with every Bob's encrypted footprint to find the number of common words. The detailed algorithm description is given in Figure 7.

Alice	Bob
1. Find the words, that are in all Alice's footprints and transform them into numerical representation to get set Com_A	1. Find the words, that are in all Bob's footprints and transform them into numerical representation to get set Com_B
For every Alice's footprint $footpr_{Ai}$ repeat step 2:	For every Bob's footprint $footpr_{Bj}$ repeat step 2:
2. Transform the footprint words, that are not in Com_A into numerical representation and add them to the set $footpr_{Ai}$	2. Transform the footprint words, that are not in Com_B into numerical representation and add them to the set $footpr_{Bj}$
3. Choose the parameters for a commutative encryption function Enc_1	3. Choose the parameters for a commutative encryption function Enc_2
Encryption public parameters →	

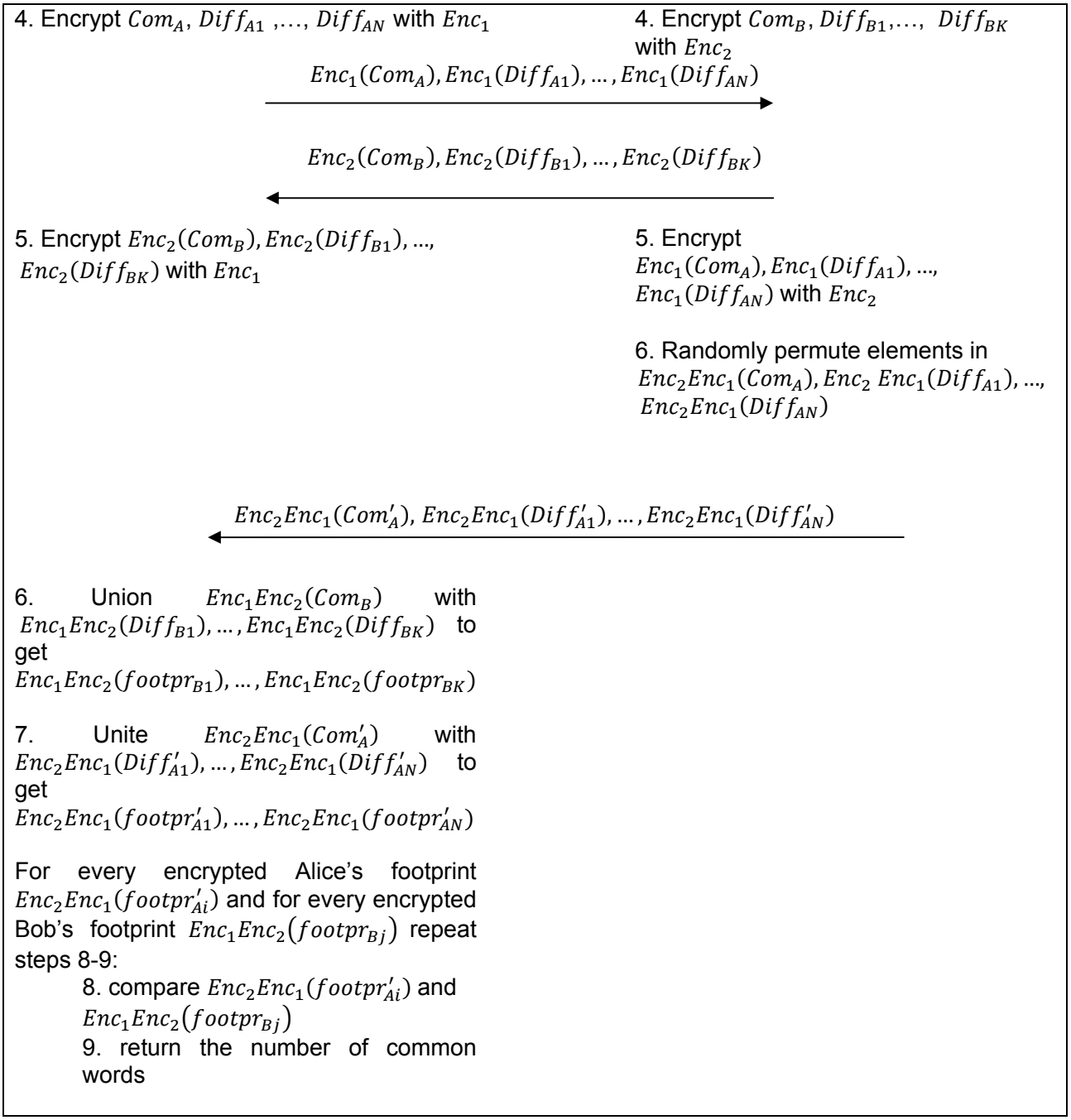


Figure 7: Method 1 footprint comparison algorithm outline

3.4.4.2 Method 2 footprint comparison algorithm

Method 2 exploits modified private-matching scheme to perform footprint comparison in privacy-preserving way. Modified private-matching scheme is a two-party protocol based on private-matching scheme described in paragraph 2.2. Unfortunately, it wasn't possible to use original private-matching scheme without any changes, because it reveals the words, containing in footprints intersection. In modified private matching scheme this drawback was eliminated. It

guarantees the privacy protection for all footprints words of every party. The description of modified private matching scheme steps can be found in Figure 9 below.

Method 2 footprint comparison algorithm is the following. Let Alice has a set of footprints $Footpr_A = \{footpr_{A1}, \dots, footpr_{AN}\}$. Bob has a set of footprints $Footpr_B = \{footpr_{B1}, \dots, footpr_{BK}\}$. As well as in Method1 the algorithm starts from the footprints preparation for comparison process. Alice divides each footprint into 2 subsets. The first one consists of the words, containing in all her footprints. The second one consists of the rest of the words, not containing in the first subset. Alice transforms every word in the subsets into numerical representation. As the result she gets subsets Com_A, $Diff_{A1}$,…, $Diff_{AN}$. The same actions are performed by Bob and he obtains the subsets Com_B , $Diff_{B1}$,…, $Diff_{BK}$. Alice defines a homomorphic encryption function and sends public parameters to Bob. Then for every of Alice's subsets the following steps are performed:

Let Alice 's subset consist of $\{x_1, \dots, x_L\}$. Alice defines a polynomial P, as shown below:

$$\mathrm{P(t)} = (\mathrm{x_1} - \mathrm{t})(\mathrm{x_2} - \mathrm{t}) \dots (\mathrm{x_L} - \mathrm{t}) = \sum_{\mathrm{u=0}}^{\mathrm{L}} \mathrm{a_u t^u}$$

She sends encrypted coefficients to Bob. Bob evaluates the polynomial at each of his numbers of each of his subsets. As the result he obtains the sets $Enc\big(P(Com_B)\big), Enc\big(P(Diff_{B1})\big), \dots, Enc(P(Diff_{BK}))$. He raises each of his result to a random power r and multiplies it by an encryption of 1 to get sets $Enc(r_0 \cdot P(Com_B) + 1)$, $Enc(r_1 \cdot P(Diff_{B1}) + 1)$,…, $Enc(r_K \cdot P(Diff_{BK}) + 1)$. He sends them to Alice. Alice decrypts received data, count the number of 1's in each of Bob's decrypted set. This number represents the number of common words, sharing by Alice's subset and corresponding Bob's subset.

The number of common words, containing in Alice's footprint $footpr_{Ai}$ and Bob's footprint $footpr_{Bj}$ for $i = \overline{1,N}$, $j = \overline{1,K}$ is calculated as follows:

$$num\_words(footpr_{Ai}, footpr_{Bj}) = num\_words(Com_A, Com_B) + num\_words(Com_A, Diff_{Bj}) + \\ + num\_words(Diff_{Ai}, Com_B) + num\_words(Diff_{Ai}, Diff_{Bj}),$$

where $num\_words(X, Y)$ is a number of common words, containing in set X and set Y

The detailed description of Method 2 footprint comparison algorithm is given in Figure 8. The overview of modified private matching scheme is represented in Figure 9.

Alice	Bob
1. Find the words, that are in all Alice's footprints and transform them into numerical representation to get set Com_A	1. Find the words, that are in all Bob's footprints and transform them into numerical representation to get set Com_B
For every Alice's footprint $footpr_{Ai}$ repeat step 2: 2. Transform the footprint words, that are not in Com_A into numerical representation and add them to the set $Diff_{Ai}$	For every Bob's footprint $footpr_{Bj}$ repeat step 2: 2. Transform the footprint words, that are not in Com_B into numerical representation and add them to the set $Diff_{Bj}$
3. Choose the parameters for a homomorphic encryption function Enc	
Public encryption parameters →	
For every set $SetA$ in { Com_A , $Diff_{A1}$,…, $Diff_{AN}$} repeat steps 4-6: 4. define polynomial P with coefficients $a_0, \dots, a_L$ 5. Encrypt coefficients, to get set $\{Enc(a_0), \dots, Enc(a_L)\}$	
$\{Enc(a_0), \dots, Enc(a_L)\}$ →	
	For every set $SetB$ in { Com_B , $Diff_{B1}$,…, $Diff_{BK}$} do steps 3-4: 3. For every element in $SetB$ repat steps a-b: a. Choose a random value r b. compute $Enc(r \cdot P(y) + 1)$ 4. randomly permute elements in set $Enc(r \cdot P(SetB) + 1)$ to get $Enc(r \cdot P(SetB') + 1)$
← $Enc(r_0 \cdot P(Com'_B) + 1), Enc(r_1 \cdot P(Diff'_{B1}) + 1),\dots, Enc(r_K \cdot P(Diff'_{BK}) + 1)$	
6. For every set $SetB$ in { $Enc(r_0 \cdot PComB'+1$, $Enc(r1 \cdot P(DiffB1')+1)$,…, $Enc(r_K \cdot P(Diff'_{BK}) + 1)$ } repeat steps a-b: a. Decrypt all values in $SetB$ b. Count number of $1's$ to get	

number of common words between $SetA$ and $SetB$, denoted n$um_words(SetA, SetB)$

For every Alice's footprint $footpr_{Ai}$ and every Bob's footprint $footpr_{Bj}$ repeat steps 7-8:

7. Find the number of common words $num\_words(footpr_{Ai}, footpr_{Bj})$
8. Return the number of common words

Figure 8: Method 2 footprint comparison algorithm outline

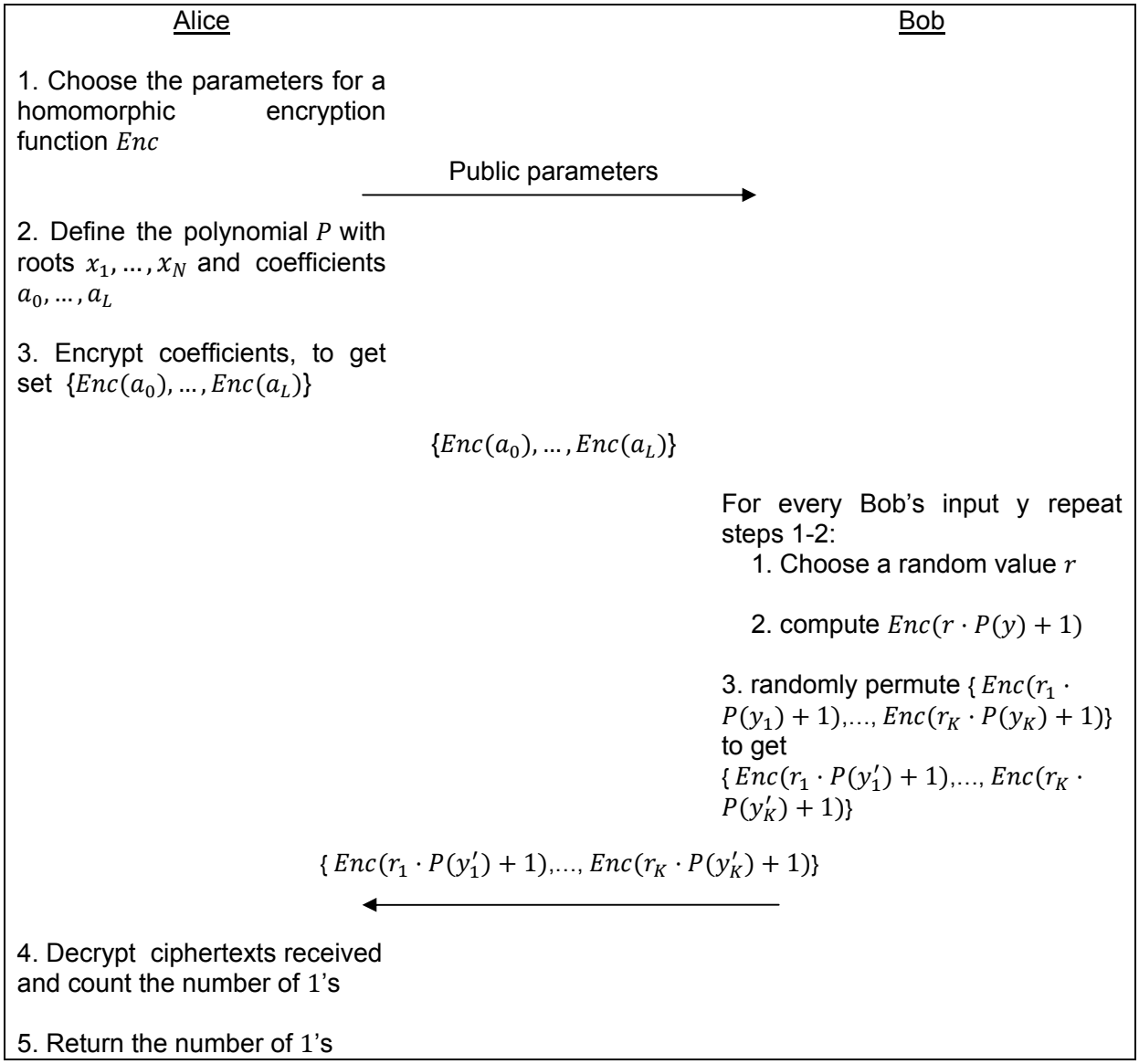


Figure 9: Modified private matching scheme overview

3.4.4.3 Word transformation into numerical representation

Both modified private matching scheme and privacy-preserving data comparison protocol operate with numbers. So in order to use these methods for footprints comparison, it is necessary to transform all words into numerical representation. This is done according to the following rule:

Each word is considered as a string of symbols$x_1, \dots, x_N$, where x_i, $i = \overline{1,N}$ can be a number, a Latin letter, or one of the symbols '-', ' ', '' (empty symbol). For each symbol the number, corresponding to this symbol, is determined according to the Table 1. Then the word is represented as a number $F(x_0x_1 \dots x_N)$, calculated by the formula below:

$$F(x_0x_1 \dots x_N) = a_0(x_N) \cdot 39^0 + a_1(x_{N-1})39^1 + \cdots + a_N(x_0)39^N,$$

where $a_i(x_{N-i})$ is a number corresponding to the symbol (x_{N-i})

Symbol	Number	Symbol	Number	Symbol	Number
0	27	d	4	q	17
1	28	e	5	r	18
2	29	f	6	s	19
3	30	g	7	t	20
4	31	h	8	u	21
5	32	i	9	v	22
6	33	j	10	w	23
7	34	k	11	x	24
8	35	l	12	y	25
9	36	m	13	z	26
a	1	n	14	' '	37
b	2	o	15	'-'	38
c	3	p	16	''	0

Table 1: Symbol encoding table

3.4.5 Area names comparison

The goal of the area names comparison is to find out the type of semantic relationship between a pair of documents. Without the information about semantic relationship the documents will be just considered as having a common part.

The types of semantic relationship are the following:

1. The area names are equal
2. The area names of one party is equal to the some area names of another party
3. The areas names are not equal

The type of semantic relationship between documents is detected by the performing of two steps. During the first step the area names are compared according to the chosen method for privacy-

preserving document similarity detection and the number of common names is found. In Method 1 privacy-preserving data comparison protocol, given in paragraph 2.3, is used. In Method 2 modified private matching scheme, described in paragraph 3.4.4.2 in Figure 9, is applied. During the second step the type of semantic relationship between documents is determined as it described below:

Let Th_1 be Alice's list of area names and Th_2 be Bob's list of area names. L_1 is a power of Th_1, L_2 is a power of Th_2 and Com is a number of common areas, founded in first step. Then:

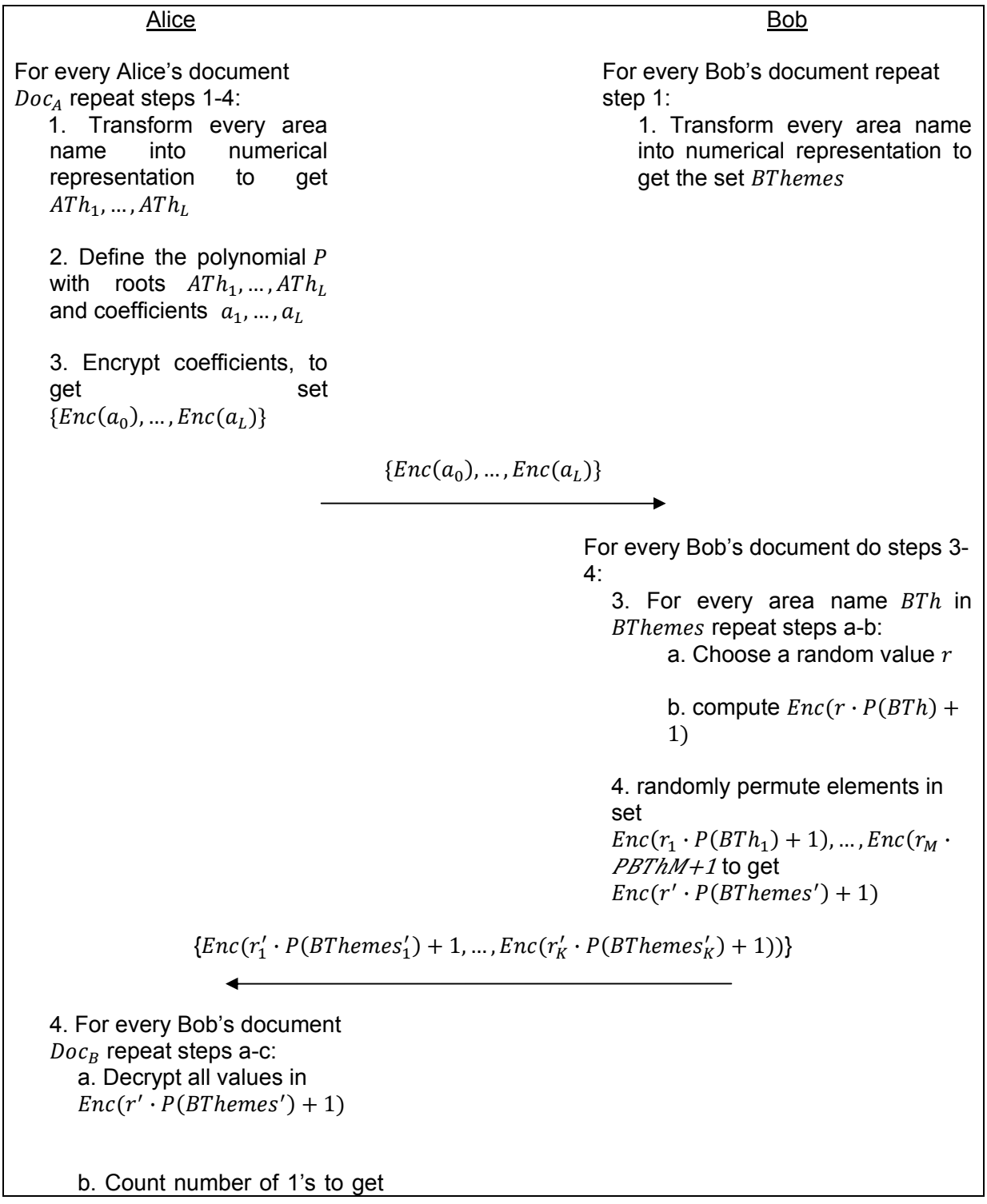
If Com==0:
Area names are not equal
else:
If $L_1 = Com$ and $L_2 = Com$:
Area names are equal
If $L_1 > Com$ and $L_2 = Com$:
Bob's area names are equal to some Alice's area names
If $L_1 = Com$ and $L_2 > Com$:
Alice's area names are equal to some Bob's area names
If $L_1 > Com$ and $L_2 > Com$:
Area names are not equal

The general description of Method 1 and Method 2 area names comparison algorithms is given in Figure 10 and Figure 11, respectively.

Alice		Bob
For every Alice's document repeat steps 1-4: 1. Transform every area name into numerical representation and encrypt with Enc_1 to get set $E_1(Th_A)$	$E_1(Th_A)$ →	For every Bob's document repeat steps 1-3: 1. Transform every area name into numerical representation and encrypt with Enc_2 to get set $E_2(Th_B)$
	← $E_2(Th_B)$	
2. Encrypt $E_2(Th_B)$ with Enc_1 to get set $E_1E_2(Th_B)$		2. Encrypt $E_1(Th_A)$ with Enc_2 to get set $E_2E_1(Th_A)$ 3. Randomly permute $E_2E_1(Th_A)$to get set $E_2E_1(Th'_A)$
	← $E_2E_1(Th'_A)$	
3. compare $E_1E_2(Th_B)$ and $E_2E_1(Th'_A)$		

4.return the number of common areas

Figure 10: Method 1 area names comparison algorithm outline



number of common area names between Doc_A and Doc_B, called Com

c. Return Com

Figure 11: Method 2 area names comparison algorithm outline

.

3.4.6 Result finding

The following types of similarity can be determined using one of the methods for privacy-preserving document similarity detection:

1. The documents are syntactically similar (duplicates)
2. The documents are almost syntactically similar (near duplicates)
3. The documents are semantically similar
4. The documents share common part
5. A is semantically similar to the some part of B
6. B is semantically similar to the some part of A
7. Documents are different

In order to find a type of similarity between pair of documents, it is necessary to calculate N_1 and N_2 according to the formulas below:

$$N_1 = \frac{number\ of\ common\ words}{number\ of\ all\ words\ in\ Alice's\ footprint}$$

$$N_2 = \frac{number\ of\ common\ words}{number\ of\ all\ words\ in\ Bob's\ footprint}$$

Such similarity metric was chosen because it allows to detect the situations when one document is a part of another one.

Then according to the N_1 and N_2 values and type of semantic relationship, the type of document similarity is determined as it shown in Table 3. The intervals for $N_1$and N_2values are preliminary and must be revised after implementation and testing phases depending on experiments results. The notations, used in Table 3 are explained in Table 2.

Notation	**Meaning**
A	Alice's document
B	Bob's document
$Th_1 = Th_2$	Area names are equal
$Th_1 \subset Th_2$	Alice's area names are equal to some Bob's area names
$Th_2 \subset Th_1$	Bob's area names are equal to some Alice's area names

$Th_1 \neq Th_2$	Area names are not equal
$A = B$ syntactically	The documents are syntactically similar (duplicates)
$A \approx B$ syntactically Near duplicated	The documents are almost syntactically similar (near duplicates)
$A = B$ semantically	The documents are semantically similar
$A \subset B$ semantically	A is semantically similar to the some part of B
$B \subset A$ semantically	B is semantically similar to the some part of A
Common part	The documents share common part
$A \neq B$	Documents are different

Table 2: Notations for Table 3

Input parameters			**Output parameters**
N1	**N2**	**Type of semantic relationship**	**Type of document similarity**
1.0	1.0	any	$A = B$ syntactically Duplicate
	[0.85;1.0)	any	$A \approx B$ syntactically Near duplicated
	[0.4;0.85)	any	$A \subset B$ syntactically
	(0;0.4)	any	$A \subset B$ syntactically
[0.85;1.0)	1.0	any	$A \approx B$ syntactically Near duplicated
	[0.85;1.0)	any	$A \approx B$ syntactically Near duplicated
	[0.4;0.85)	$Th_1 = Th_2$	$A = B$ semantically
		$Th_1 \subset Th_2$	$A \subset B$ semantically
		$Th_2 \subset Th_1$	$B \subset A$ semantically
		$Th_1 \neq Th_2$	Common part
	(0;0.4)	$Th_1 = Th_2$	$A \subset B$ semantically
		$Th_1 \subset Th_2$	$A \subset B$ semantically
		$Th_2 \subset Th_1$	$B \subset A$ semantically
		$Th_1 \neq Th_2$	Common part
[0.4;0.85)	1.0	any	$B \subset A$ syntactically
	[0.85;1.0)	$Th_1 = Th_2$	$A = B$ semantically
		$Th_1 \subset Th_2$	$A \subset B$ semantically
		$Th_2 \subset Th_1$	$B \subset A$ semantically
		$Th_1 \neq Th_2$	Common part
	[0.4;0.85)	$Th_1 = Th_2$	$A = B$ semantically
		$Th_1 \subset Th_2$	$A \subset B$ semantically

		$Th_2 \subset Th_1$	$B \subset A$ semantically
		$Th_1 \neq Th_2$	Common part
	(0;0.4)	any	$A \neq B$
(0;0.4)	1.0	any	$B \subset A$ syntactically
	[0.85;1.0)	$Th_1 = Th_2$	$B \subset A$ semantically
		$Th_1 \subset Th_2$	$A \subset B$ semantically
		$Th_2 \subset Th_1$	$B \subset A$ semantically
		$Th_1 \neq Th_2$	Common part
	[0.4;0.85)	any	$A \neq B$
	(0;0.4)	any	$A \neq B$
0	0	any	$A \neq B$

Table 3: Documents similarity type finding table

3.5 Implementation

In this paragraph the implementation step is given in details. In contrast to the design part, the description is presented separately for each method.

3.5.1 Method 1 implementation

Method 1 was implemented according to the design specification described in paragraph 3.4 in the following way. Let Alice and Bob represent two parties. Alice is responsible for finding the similarity types between documents. Method 1 starts with the procedure that checks if both parties have documents to compare. The next step is a knowledge base preparation. During this step each party gathers together all necessary data for footprint and area finding steps execution. Having prepared the knowledge base, Alice and Bob find areas and footprints for their documents. If some document contains less than 20 words, this document doesn't participate in comparison process any more. It's prohibited to compare small size footprints in order to eliminate the opportunity to reveal the words of the other party footprints. If at least one party has all footprint containing less than 20 words, Method 1 execution is stopped. Else method execution is proceeding and parties send the lists of their document names to each other. After that Alice and Bob choose the parameters for their Massey-Omura encryption functions. Then footprints are compared and Alice gets the number of common words between each of her footprints and every Bob's one. According to these results, she finds the types of similarity between their documents.

The general overview of all Method 1 implementation steps is given in Figure 12. The detailed overview of each step can be found below.

1. Check if Alice and Bob have documents to compare. If they have, go step 2, else STOP
2. Each party performs knowledge base preparation step
3. Each party finds areas and footprint for every document
4. Each party deletes all documents, whose footprints contain less than 20 words
5. Check if Alice and Bob have documents to compare. If they have, go step 6, else STOP
6. Each party sends the list of the document names to another party

7. Footprint comparison
8. Alice finds the types of similarity between documents

Figure 12: Method 1 implementation steps overview

3.5.1.1 Knowledge base preparation

Several databases were used to form a knowledge base. The first one, called in the report as NLTK database, consists of the databases proposed by Natural Language Toolkit (NLTK). NLTK is an open source library that can be freely downloaded from [29]. "It was originally created in 2001 as a part of a computational linguistics course in the Department of Computer and Information Science at the University of Pennsylvania.
Since then it has been developed and expanded with the help of dozens of contributors. It has now been adopted in courses in dozens of universities, and serves as the basis of many research projects." [30]. NLTK database is used for part of speech tagging and stemming.

The second database, called PoS (Part of Speech) database, was created to improve the part of speech tagging process and to be used for deleting of the frequently used words. It consists of seven .txt files. The file names and their description can be found in Table 4. The content of all files can be found in Appendix A. The stop words and phrases were taken from [31], [32], [33], the prepositions - from [34], [35], [36], and adverbs – from [37], [38], [39].

File name	Description
stopwords_mult.txt	Contains stopwords, that are phrases
stopwords_one.txt	Contains stopwords, that are single words
preposit_mult.txt	Contains prepositions, that are phrases
preposit_one.txt	Contains prepositions, that are single words
adverb_mult.txt	Contains adverb, not ending with –ly, that are phrases
adverb_one.txt	Contains adverb words, not ending with –ly, that are single words
nonadverb.txt	Contains the words, ending with –ly, that are not adverbs

Table 4: PoS database files

Another data base, called Ontology database, was created for ontology implementation. This database consists of 3 files: ontology_structure.txt, ontology_mult.txt, ontology_one.txt. The first file contains information about ontology areas names, and number of terms, describing these areas. The second file contains ontology term phrases, and the third one contains ontology single words terms. These files have fixed structures. They are the following:

*ontology_structure.txt structure***:**

File_line::=Area':'Name_parent':'Num_area':'Synonym*['','Synonym]*
Area::=String
Name_parent::=String
Num_ area::=NUM+
Synonym::=String
NUM::='0'|-|'9'

Area is a name of area,
Name_parent is a name of ancestor area for Area
Num_area is a number of terms, describing Area
Synonym is a synonym for Area

ontology_mult.txt structure:

File_line::=Term':' Area
Term::=String
Area::=String

Term denotes the ontology term, consisting of two or more words.
Area is a name of area, described by Term

ontology_one.txt structure:

File_line::=Term':' Area
Term::=String
Area::=String

Term denote the ontology term, consisting of single word.
Area is a name of area, described by Term

For ontology implementation the computer security domain was chosen. For the testing purpose only the authorization area was processed. Authorization area contains the role-based access control, mandatory access control, discretionary access control and multilevel access control areas. Multilevel access control consists of Bel-Lapadula area. For formation of the ontology term sets, the following sources were used: [47], [48], [49]. The ontology object diagram is presented in Figure 13.The content of ontology database can be found in Appendix B.

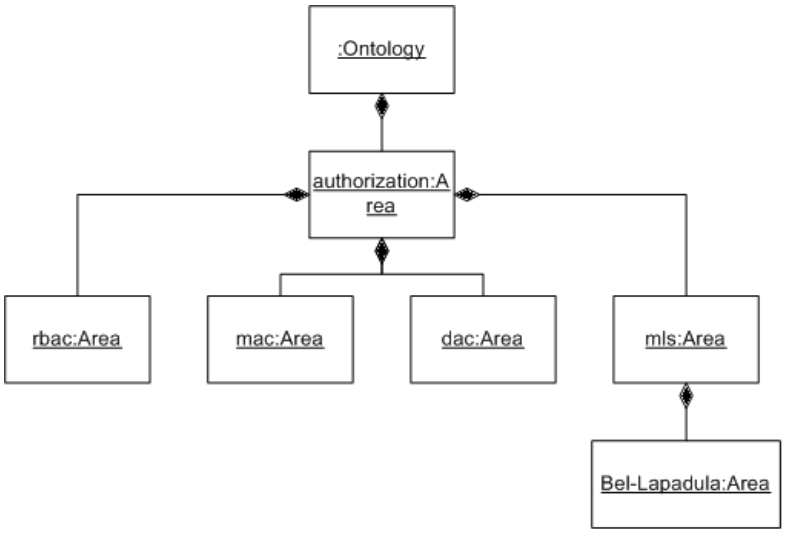


Figure 13: Ontology object diagram

The knowledge base structure overview is given below in Figure 14.

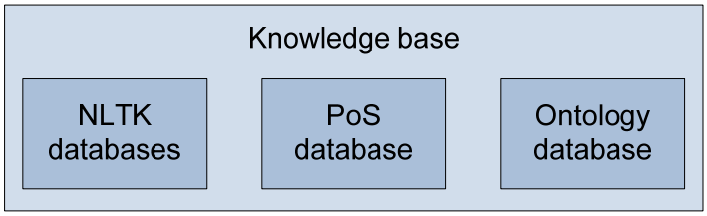


Figure 14: Knowledge base overview

In order to prepare the knowledge base the following steps should be performed:

1. Load stop words : phrases, words from stopwords_mult.txt, stopwords_one.txt
2. Load prepositions: phrase, words from preposit_mult.txt, preposit_one.txt
3. Load adverbs: phrases, words from adverb_mult.txt, adverb_one.txt
4. Load not adverbs, ending with –ly from nonadverb_ly.txt
5. Load data base from NLTK module
6. Load ontology terms: phrases, words from ontology_mult.txt, ontology_one.txt
7. Load ontology structure from ontology_structure.txt

3.5.1.2 Area and footprint finding

The area and footprint finding steps are implemented according to the description given in paragraphs 3.4.2 and 3.4.3, respectively. During the implementation these steps were united together for optimization of the finding process.

In order to find areas and footprint the following steps are implemented:

1. Make all letters to be lower-case
2. Delete all end of line symbols ‘\n’
3. For every phrase in the list of ontology term phrases, check if it is in the document. If it is, then delete it from the document, and add to the list of document terms without stemming and to the footprint after stemming
4. For every phrase in the list of preposition phrases, check if it is in the document. If it is, then delete it from the document.
5. For every phrase in the list of adverb phrases, check if it is in the document. If it is, then delete it from the document.
6. For every phrase in the list of stop words phrases, check if it is in the document. If it is, then delete it from the document.
7. Tokenize the reduced document. Tokens shouldn’t include any symbols or be the symbol. Composite adjective should be divided into 2 parts. No repetitions are allowed.

8. For every word in the list of ontology terms, check if it is in the token list. If it is, then delete it from the token, and add to the list of document terms without stemming and to the footprint after stemming, if it hasn't been added before.
9. For every token, check if it is in the list of preposition words. If it is, then delete it from the token list
10. For every token, check if it is in the list of stopwords. If it is, then delete it from the token list
11. For every token check if it is an adverb :
    a. Check if it is in the list of adverb words .If it is, then delete it from the token list
    b. Check if it ends with –ly. If it does and it is in the list with not adverbs words, ending with –ly, then add this token to the footprint after stemming . Delete token from the token list .
12. For every token find its part of speech.
13. Delete all tokens, except nouns, adjectives and verbs
14. Make stemming for every token
15. Delete repetitions in stemming token list and add them to the footprint list. Footprint is ready
16. For every found ontology term find the areas it describes.
17. For every found area Th do following:
    Let $terms_{Th}$ be the number of founded in document terms describing the area Th, and All_terms_{Th} be the number of ontology terms in term set, describing area Th.

    Find $E = \frac{terms_{Th}}{All\_terms_{Th}}$

    If E< 0.1, then document contents doesn't related to the area Th. Otherwise add its name to the list of document area names.

Stemming, part of speech tagging and tokenization were performed by functions $stem()$, $pos\_tag()$, and $word\_tokenize()$, respectively, implemented in NLTK module.

3.5.1.3 Footprint comparison

Footprint comparison was implemented exactly as it was described in paragraph 3.4.4. In order to encrypt footprint words and area names the Massey-Omura cryptosystem was applied. It was implemented in the following way:

Key generation:

1. Bob generates public key q, that is a random positive prime 1024 bit long number.
2. Alice selects public key e_1, a random positive prime 256 bit long number and computes $d_1 = e_1^{-1} mod\ q - 1$.
3. Alice selects public key e_2, a random positive prime 256 bit long number and computes $d_2 = e_2^{-1} mod\ q - 1$.

The usage of prime numbers as a public keys e_1, e_2 guaranties that conditions $gcd(e_1, q-1) = 1$ and $gcd(e_2, q-1)$ are met. For finding large prime number the function $getPrime()$ from standard python module "number", was used. To find d_1 and d_2, the function $inverse()$, containing in the same module, was utilize.

Encryption:

Let m to be a number to be encrypted. Then encryption of m will be calculated as $c = m^e mod\ q$, where e is Alice's or Bob's private key.

Unfortunately, it wasn't possible to use standard python function $pow()$ for encryption, because it can't operate with a numbers of such long size. So the new procedure was created. It calculates the modulus of the numbers of any length and is based on the modulus properties and function $pow()$. The following modulus properties were used:

1. If b is odd, then $a^b mod\ N = (a\ mod\ N * a^{b-1} mod\ N)\ mod\ N$.
2. If b is even, then $a^b mod\ N = (a^{\frac{b}{2}}\ mod\ N)^2 mod\ N$

The code of procedure for finding $a^b mod\ N$, is presented below in Figure 15

```
def large_modulus(a,b,N):

    m=1
    while b>0:
        if not b%2==0:
            m=m*a
            m=pow(m,1,N)
        b=b/2
        a=pow(a*a,1,N)

    return m
```

Figure 15: Modulus finding procedure

3.5.1.4 Result finding

Since it is not always required to find the semantic relationship between documents, the area names comparison step is executed only when it is needed. Decision if semantic relationship finding is needed is made during the result finding step.

Result finding step was implemented as it is described in paragraph 3.4.5. The area name comparison step is implemented according to the description given in paragraph 3.4.5 in Figure 10.

3.5.2 Method 2 implementation

Method 2 was implemented according to the description given in paragraph 3.4. It has the same steps, as Method 1 has. They can be found in paragraph 3.5.1 in Figure 12. These steps, except of footprint and area names comparison ones, are implemented in the same way as it were done for Method 1 (see paragraph 3.5.1). The description of footprint and area names comparison steps implementation is given below.

3.5.2.1 Footprint comparison

Footprint comparison was implemented according to the algorithm, given in Figure 8 in paragraph 3.4.4.2. This algorithm is based on the usage of homomorphic encryption. In Method 2 Paillier's cryptosystem was used. It was implemented as follows:

Key generation

1. Choose two 512 bit long prime numbers p and q
2. Set $n = p \cdot q$
3. Set $g = n + 1$
4. Consider n, g as public parameters and p, q as private ones.

Encryption

1.Let m be a plaintext to be encrypted, where $m \in \mathbb{Z}_n^*$
2. Select a random 1023 bits long number r
3. Compute ciphertext $c = g^m \cdot r^n \bmod n^2 = (g^m \bmod n^2 \cdot r^n \bmod n^2) \bmod n^2$

Decryption

1. Given ciphertext $c \in \mathbb{Z}_{n^2}^*$
2. Compute plaintext $= L(c^\lambda \bmod n^2) \cdot \left(L(g^\lambda \bmod n^2)\right)^{-1} \bmod n$, where $L(u) = \frac{u-1}{n}$ and $\lambda = \frac{(p-1)\cdot(q-1)}{GCD(p-1,q-1)}$

The functions, applied for prime numbers finding, calculating of the modulus of large numbers and multiplicative inverse to the modulus are the same as in Massey-Omura Cryptosystem implementation description in paragraph 3.5.5.3. Function $GCD()$ from standard python module "number" was used to find the great common divisor.

In order to use Paillier's cryptosystem, the modified private matching scheme was adapted. Instead of comparison of all elements the set at once, the set is divided into subsets, so that the multiplication of all elements in every subset is less than public key n. This is done to be sure that all coefficients are less than n and will be encrypted in a proper way. The number of common elements between two sets is a summation of number of common elements between each subset of one party and set of another party.

The footprint comparison step was implemented in such way that while one party is finding the polynomial values, the other party finds the coefficient of the new polynomial and decrypts the

values of the previous one. The simultaneous performance of operations by parties reduces the execution time significantly.

The overview of footprint comparison implementation step is given in Figure 16. The description of function for any two sets comparison is presented in Figure 17.

Alice		Bob
1. Find the words, that are in all Alice's footprints and transform them into numerical representation to get set Com_A		1. Find the words, that are in all Bob's footprints and transform them into numerical representation to get set Com_B
For every Alice's footprint $footpr_{Ai}$ repeat step 2:		For every Bob's footprint $footpr_{Bj}$ repeat step 2:
2. Transform the footprint words, that are not in Com_A into numerical representation and add them to the set $Diff_{Ai}$		2. Transform the footprint words, that are not in Com_B into numerical representation and add them to the set $Diff_{Bj}$
3. Choose the parameters for a homomorphic encryption function Enc		
	Public encryption parameters →	
For every set $SetA$ in { Com_A , $Diff_{A1}$,…, $Diff_{AN}$ } repeat steps 4:		
4. find the number of common elements between $SetA$ and every set from the { Com_B , $Diff_{B1}$,…, $Diff_{BK}$}, according to the algorithm in Figure 17	↔	3. help Alice to compare Alice's setA and Bob's set { Com_B , $Diff_{B1}$,…, $Diff_{BK}$ }, according to the algorithm in Figure 17

Figure 16: Method 2 footprint comparison implementation description

Alice	Bob
1.Start position $St = 0$	1.Receive Enc_Coef
2.Find the index End of $SetA$ element, such that $el_{St} \cdot \ldots \cdot el_{End} < N$ and $el_{St} \cdot \ldots \cdot el_{End+1} > N$	For every element y in every set $\{Set_{B1},..,Set_{BK}\}$, do steps 2-3:
3. Define polynomial P with set of coefficients $Coef$ with roots $el_{St}, \ldots, el_{End}$	2. generate random number r in range (5;35)
4. Encrypt coefficients to get set Enc_Coef	3. Find the value of $Enc(r \cdot P(y) + 1)$

5.Send Enc_Coef

6. If $St = L$, do steps 7-10, else go to the step 11:

7. Receive encrypted values of polynomial with coefficients $a_1, \dots, a_{St-End}$ form Bob

8. send “done” to Bob

9.Decrypt values and count the number of 1’s for every Bob’s set separately

10. Return the number of 1’s for every set

11. for $i = \overline{0,K}$ Num_one_i=0

12. while $End \neq L$, repeat steps 13-23:

13. $St = End$

14.$Coef\_old = Coef$

15. Find the new index End of $SetA$ element, such that $el_{St} \cdot \dots \cdot el_{End} < N$ and $el_{St} \cdot \dots \cdot el_{End+1} > N$

16. Define polynomial P with set of coefficients $Coef$ with roots $el_{St}, \dots, el_{End}$

17. Encrypt coefficients to get $Encr\_Coef$

18.Receive encrypted values of polynomial with coefficient $Coef\_old$

15.send “more”

15. send $Encr\_Coef$

16. Decrypt values of polynomial with coefficient $Coef\_old$ and count the number of 1’s for every Bob’s set separately to get $\{num_0, num_1, .., num_K\}$

19. $i = \overline{0,K}$ $Num_{one_i} = Num_{one_i} + num_i$

20. If $End = L$, do steps a-c, else go to the step 11

4. Send values Val

5. data =”more”

While data is not equal to ”done”, do steps 6-7:

6.Receive data

7. If data not =”done” do steps 8-10:

8.Receive Coef

9.For every element y in y in every set $\{Set_{B1}, .., Set_{BK}\}$ do step a-b:

a. generate random number r in range (5;35)

b. Find the value of $Enc(r \cdot P(y) + 1)$

10. Send all values

a. receive encrypted values of polynomial with coefficients $Coef$
b. Decrypt values of polynomial with coefficient $Coef\_old$ and count the number of 1's for every Bob's set separately to get $\{num_0, num_1, .., num_K\}$
c. $i = \overline{0, K}$ $Num_{one_i} = Num_{one_i} + num_i$
d. return Num_one

Figure 17: Sets comparison algorithm description

The polynomial coefficients are found using the function find_coeff(). Its implementation is given in Figure 18

```
def find_coeff(number_list):
    n=len(number_list)
    coeff=[0,0]

    M1=[-1*number_list[0],1]
    M2=[0,0]
    for k in range(1,n):
        M2=[-1*number_list[k],1]
        for t in range(k+1):
            coeff[t]=0
        coeff.append(0)

        for i in range(0,k+1):
            for j in range(0,2):
                coeff[i+j]+=M1[i]*M2[j]

        M1.append(0)

        for t in range(k+2):
            p=coeff[t]
            M1[t]=p

    return coeff
```

Figure 18: find_coeff() function implementation

Let polynomial $\mathrm{P(t)} = (\mathrm{x_1} - \mathrm{t})(\mathrm{x_2} - \mathrm{t}) \ldots (\mathrm{x_L} - \mathrm{t}) = \sum_{\mathrm{u}=0}^{\mathrm{L}} \mathrm{a_u t^u}$ and N to be a public key of Pailier's cryptosystem. Then value of expression $Enc(r \cdot P(y) + 1)$ is found according to the following formula:
$Enc(r \cdot P(y) + 1) = Enc\big(P(y)\big)^r \cdot Enc(1) mod\ N^2$, where
$Enc\big(P(y)\big) = \left(\prod_{i=0}^{L} Enc(a_i)^{y^i}\ mod\ N^2\right) mod\ N^2$ and $Enc(1) = Enc(a_L)$

3.5.2.2 Area names comparison

During the result finding the type of semantic relationship between documents may be required. The area names comparison is based on the algorithm described in Figure 11 in paragraph 3.4.5 and the algorithm for sets comparison presented in Figure 17 in paragraph 3.5.2.1. The description of area names comparison algorithm implementation is given in Figure 19.

Alice		Bob
For every Alice's document repeat step 1-3:		For every Bob's document repeat step 1-2:
1. Transform area names into numerical representation to get set $Theme_A$		1. Transform area names into numerical representation to get set $Theme_B$
2.Compare $Theme_A$ with every set $\{Theme_{B1},\ldots,Theme_{BK}\}$ to get the number of common area names according to algorithm in Figure 17	⟷	2. help Alice to compare $Theme_A$ and sets $\{Theme_{B1},\ldots,Theme_{BK}\}$ according to the algorithm in Figure 17
3.Return the number of common area names		

Figure 19: Method 2 area names comparison implementation description

3.5.3 Non-privacy document similarity detection method implementation

The algorithm for non-privacy document similarity detection, called non-privacy method, was implemented to find out in how many times the performance of methods for privacy-preserving document similarity detection will increase in comparison with non-privacy method.

Non-privacy method has the same steps as Methods 1. They can be found in paragraph 3.5.1 in Figure 12. The only difference is that for comparison of footprints and area names secure protocol isn't used.

3.6 Testing

Different experiments were executed to estimate performance and to check functionality of Method 1 and Method 2. The description of the testing process is given in this chapter. The obtained results are interpreted in Discussion.

3.6.1 Testing environment description

CPU Intel Celeron (32bit) 2GHz, RAM 1GB
OS Ubuntu Linux 10.04
Python version 2.6.5
Both peers were run on the same machine.
For execution time calculation standard python function time() was used. Execution time was checked only for the peer who finds the type of document similarity.

3.6.2 Testing data

41 articles were selected from ACM [50] and IEEE [51] digitals libraries. Their sizes varied between 200 and 1000 words. Articles are related to the different areas, such as role-based access control, encryption, text mining, document similarity detection and so on.

3.6.3 Method 1 performance measurement

All experiments were done with aim to estimate performance and to check functionality. The obtained results are given below.

1. Since comparison execution time depends on the size of footprints, the series of tests were performed to compare footprint size with original document size. Their results are presented in Figure 20.

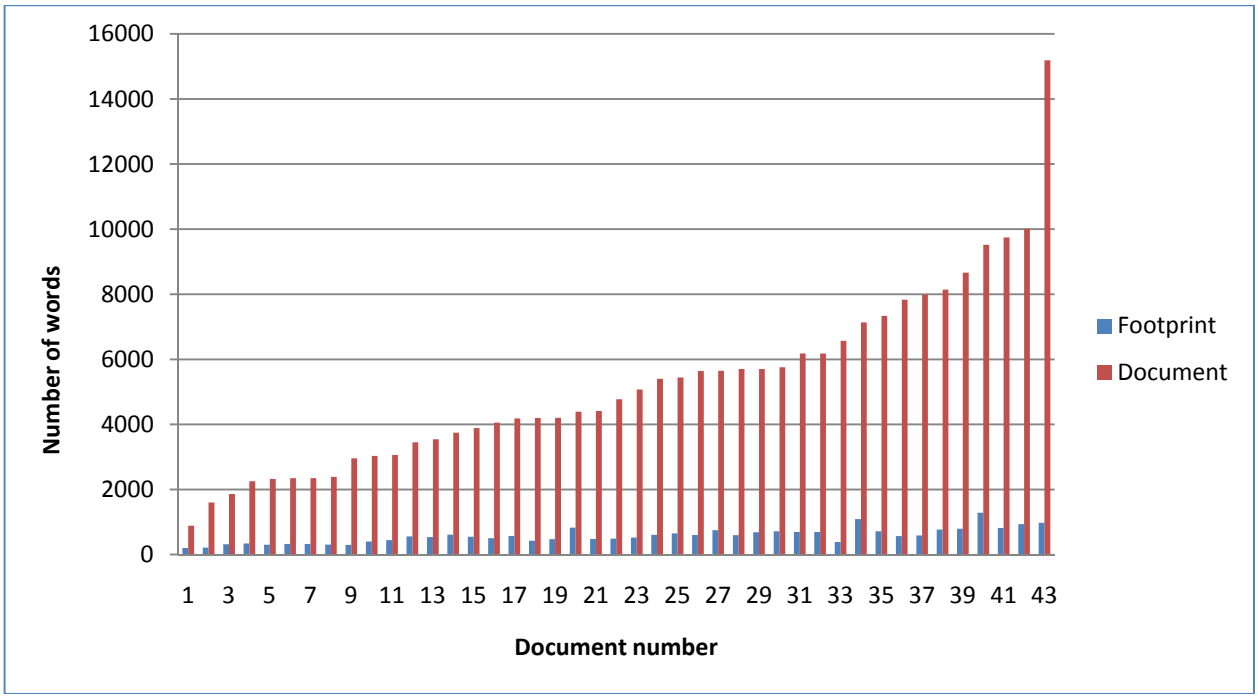


Figure 20: Footprint size versus original document size

2. Another experiments were performed to measure the time required to find footprint depending on document size. Their results are presented in Figure 21.

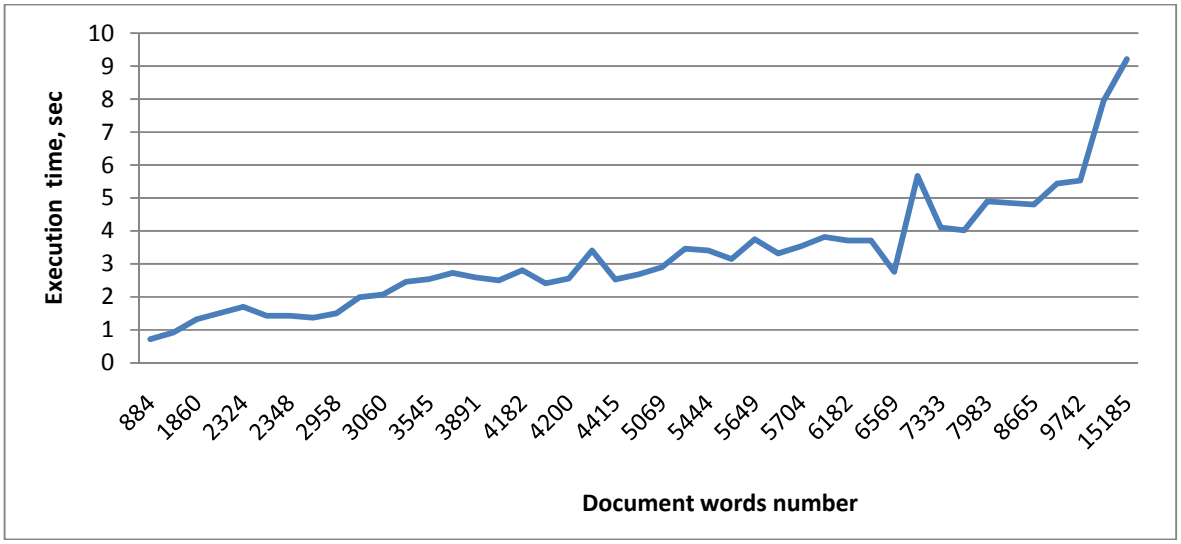


Figure 21: Footprint finding execution time

3. The next experiments were done to compare Method 1 and Non-private method performances. The results are given in Figure 22.

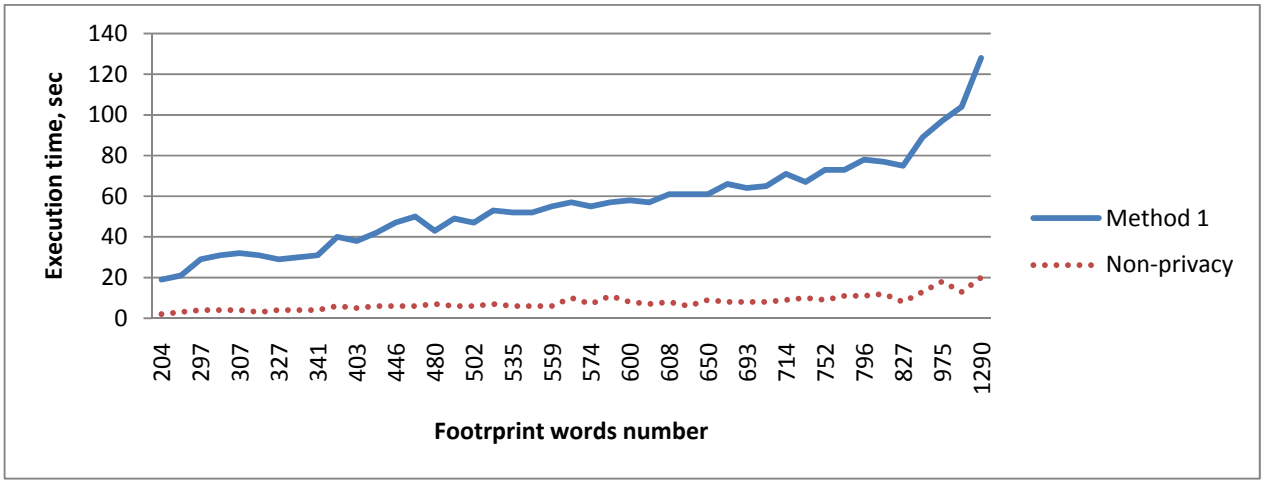


Figure 22: Method 1 performance versus Non-private method performances

4. Also there were some experiments whose aim was to compare the execution time needed to find the type of similarity between duplicates and different documents with the same size of footprints as duplicates have. It was done to be sure that the Method 1 performance doesn't depend on the type of document similarity. The results are represented in Figure 23.

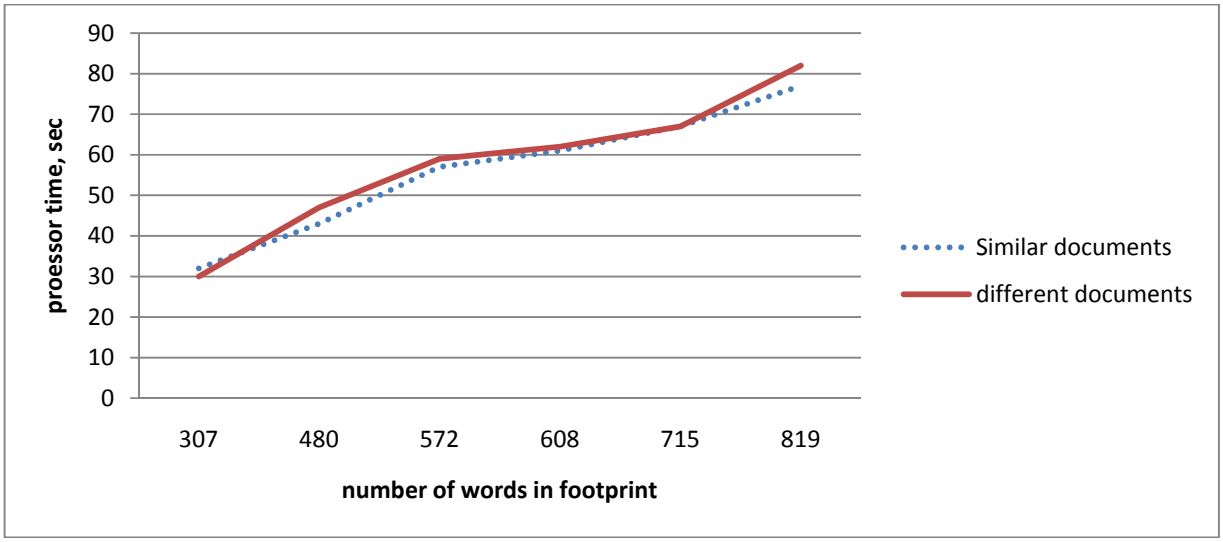


Figure 23: Comparison of time required to detect the similarity type between duplicates and different documents

5. Also some experments were performed to check if performance was improved because of dividing each footprint into two subsets (subset with the words, contains in every footprint of one party and the one with rest footprint words). This were done by the comparisson of the execution time, required to compare one Bob's documents with several Alice's documents at once or seperately for each document pair. The results are shown in Figure 24.

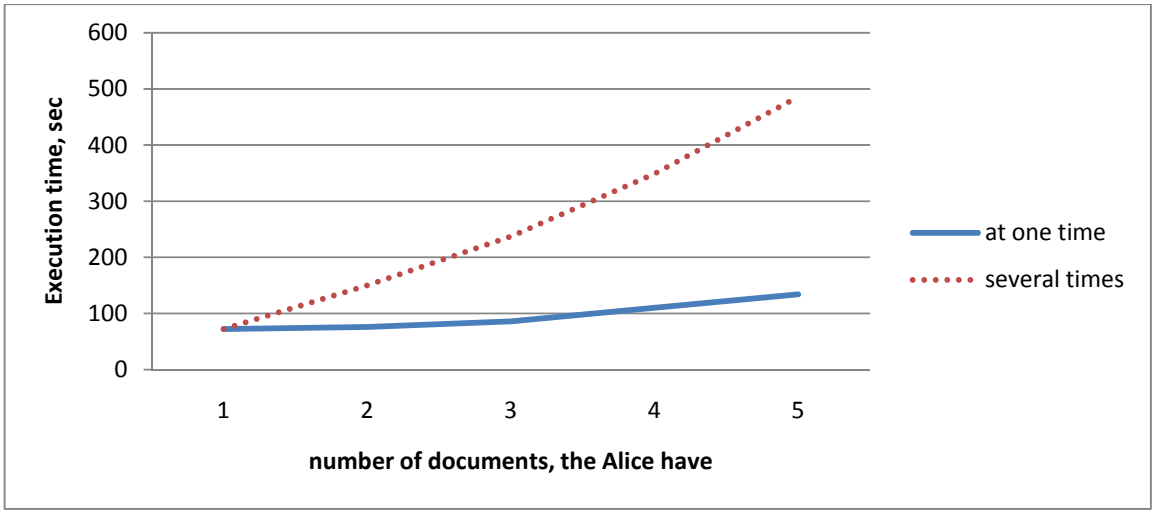


Figure 24: Comparison of time required to compare Bob's document with several Alice's documents at once or separately for each document pair.

6. The aim of these experiments was to find out how the Method 1 performance increases if the number of documents of each party grows proportionally. The results are given in Figure 25

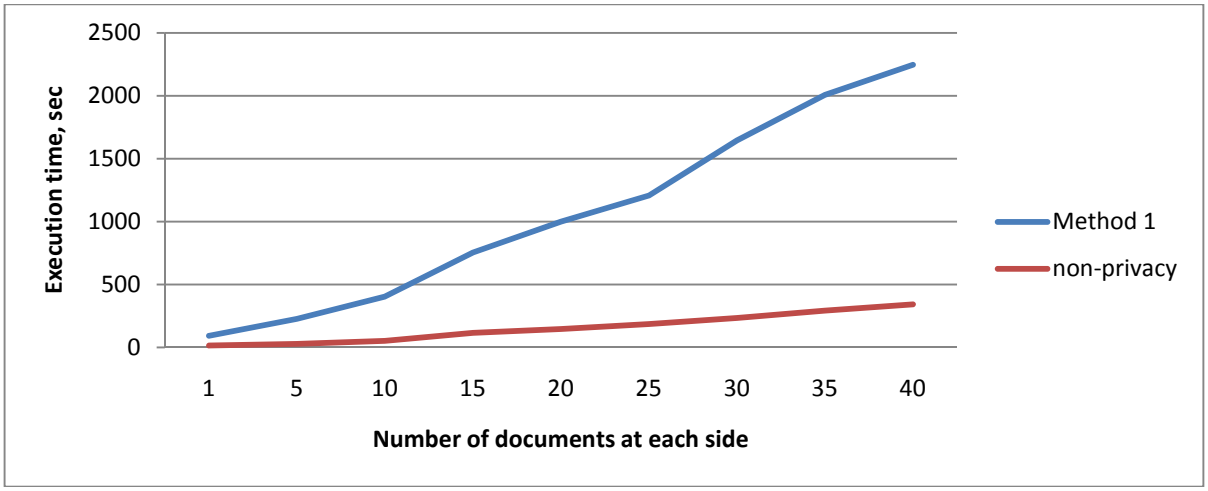


Figure 25: Method 1 performance versus the number of documents at each side.

3.6.4 Method 2 performance measurement:

1. Several experiments were performed to measure Method 2 performance. The results are presented in Figure 26.

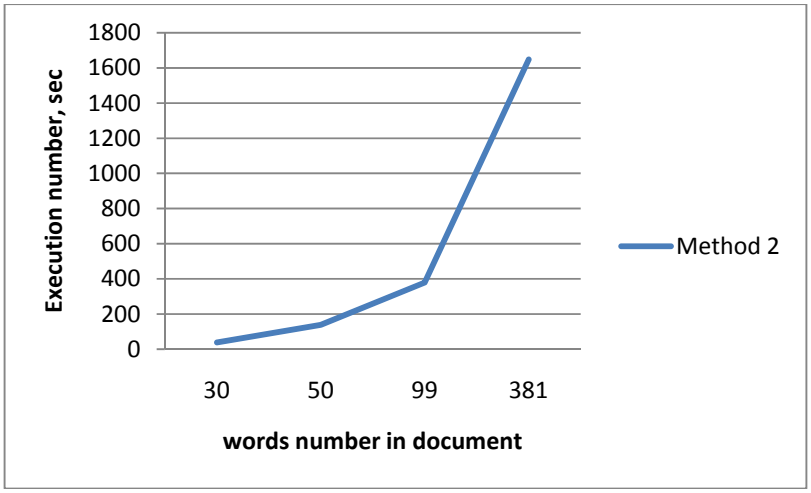


Figure 26: Method 2 performance

The execution time was unreasonable long therefore no more experiments were performed with the Method 2

3.6.5 Functionality validation

In order to validate the methods functionality 41 articles were pairwise compared. The list of document pairs together with short description is given in Table 5. The obtained type of similarity is presented in Table 6.

Number of pair	Expected type of similarity	Articles description
1	Different	Both articles have the same topic. Authors analyzing the same articles. But they are written in absolutely different ways
2	Different	Both articles have the same topic. But their contents are different
3	Semantically similar (since RBAC area is contained in ontology)	The articles have the same structures across all documents. There are identical parts of text as well as rephrased ones. In the article the same RBAC system is discussed, but authors use different names for the system components.
4	different	Authors discuss absolutely different algorithms for duplicates finding
5	Common part (because articles topic is not in ontology)	The articles have the same author. The article [7] is a short version of [10].
6	Different (since the common part is too short to detect any similarity)	The abstract and little part in the introduction is identical. The rest of the documents are absolutely different.
Other pairs	Different	All other articles are different.

Table 5: Document pairs participating in functionality validation process

Document pair	Expected type of similarity	Obtained type of similarity with ontology usage
1	Different	Common part
2	Different	Common part
3	Semantically similar	Semantically similar
4	Different	Common part
5	Common part	Common part
6	Different	Different
Other pairs	Different	Different

Table 6: Obtained types of similarity

4 Discussion

In a scope of this project the attempt were made to develop a method for identification of both syntactically and semantically similar documents in a privacy-preserving way. Documents are analysed using the ontology and main document subject areas are identified. Obtained subject areas are securely compared to determine semantic relationship between corresponding documents. Then documents are transformed into the sets of distinct meaningful words, called footprints and these sets are compared in privacy-preserving way. The type of similarity between two documents is determined according to the number of common words in footprints and documents semantic relationship The usage of the footprints but not the complete original document for documents comparison leads to the significant execution time reduction. But such approach has a drawback. It's not possible to determine if two compared documents are absolutely similar or one document was obtained from another one by making inessential changes, such as substitution of some words with synonyms, or changing the words order. To distinguish such cases, additional word-by-word comparison may be needed. As the area of application of document comparison tools (plagiarism detection and etc.) assumes that most of the documents are different, such additional word-by-word comparison step is needed rarely and thus doesn't lead to significant performance degradation.

To transform the document into the footprint some Natural Language Processing techniques were applied. There are a lot of tools for NLP processing available, for example [40],[41],[42]. In this project the natural language toolkit [29] was used. It was chosen because it's distributed under open source license, has large knowledge base and good documentation. Unfortunately, it doesn’t always determine the part of speech correctly, therefore some additional measures were taken to guarantee that method works in a proper way.

To find the documents subject areas the ontology was used. For this project only the ontology for authorization area of computer security domain was developed. But it can be easily extended with the other areas or domains. The main difference of this project ontology from the existing ones is that it is focused on the terms and what areas they describe, rather than on the areas of computer security domain and relations between them, as it is done in ontologies, described in [43], [44], [45], [46]. Real tool for the document comparison will use large ontology database and of course it will worsen the tool performance. But the subject area searching time can be reduced due to the fact that in a real use case the documents from the same domain most likely to be compared. It doesn't take sense to use astrophysics ontology comparing the anatomy articles.

Several approaches were considered for privacy-preserving footprints and subject areas comparison. They are vector space model, fingerprints and private-matching scheme. It was decided to use the private - matching scheme in this project. The vector space model was refused because it doesn't allow to detect the situation, when one document is a part of another one. The disadvantage of fingerprint approach is that it is very sensitive to the word order and, thus not

suitable for finding near duplicates. Such documents will be considered to be different. The private matching scheme doesn't have such disadvantages. It enables to identify all situations listed in paragraph 3.1. Also it is rather uncommon approach that has never been applied to the document comparison. It was interesting to evaluate its performance in such application. The drawback of private matching scheme is that it requires a lot of calculations in order to get the number of common words.

As the alternative way for secure comparison, original privacy preserving comparison protocol utilising the properties of commutative encryption, was designed. If each party has several documents for comparison than this protocol requires much less data transfers comparing with private-matching scheme. Additional time, consuming by calculation apart from the encryption, are not required. The method, based on this protocol, was expected to be rather fast and effective.
As the result, two methods for privacy-preserving document similarity detection were designed, implemented and tested. The testing showed that Method 2, based on modified privacy-preserving scheme, is very slow and isn't suitable for documents comparison. Execution time grows very fast, because it is necessary to divide the footprint into different parts to be able to encrypt the coefficients correctly. Method 1, based on privacy-preserving data comparison protocol, shows a good performance. It was experimentally determined that its execution time depends on the size of footprints and the number of documents of each party and has linear growth. Also It was proved (Figure 24) that separate handling of the common part of the footprints significantly speeds up Method 1 execution.

As for detecting of similarity types between documents, the Method1 has done it correctly, except of two cases. The first case was described in first paragraph of this chapter. The second case is following. When documents are from the same narrow area but contain different information, they considered as having common part. The reason of that is that the meaningful words are almost the same although they describe different things. To deal with it, the additional word-by-word comparison of documents is required. The wrong detection of similarity type showed that it is possible to apply this approach for privacy-preserving document clustering.

In order to prevent the revealing of the document contents during the comparison process, two approaches were applied. Both modified private matching scheme and privacy-preserving data comparison protocol guarantee the privacy of the data transfers, if the parameters of cryptosystems, used in them, are chosen correctly. During the implementation of Paillier's and Massey-Omura cryptosystems the size for public and private keys were selected in such way that the decryption of the transfer data will be practically useless. The developed methods don't required the presence of third party but doesn't guarantee the protection against man-in-the-middle attacks. So the secure connection should be established between peers. In order to provide privacy protection against malicious parties, the limitation for minimum footprint number was introduced.

5 Conclusion

The goal of this project was to develop a method for privacy-preserving document similarity detection. It should identify either semantically or syntactically similar documents. As the result two methods were developed. Both of them have the following structure. At first, the areas the documents are related to are found. Then documents are transformed into the set of distinct meaningful words. These sets as well as documents subject areas are compared in a secure way. In the first method the original privacy-preserving data comparison protocol was used for secure comparison. In the second method the modified private-matching scheme was used for same purpose. Based on the comparison results the type of similarity between documents is identified. Both of the methods provide privacy protection for the documents content of the parties.

Currently, based on testing results, the following types of document similarity are detected: near duplicates, one document is semantically or syntactically contained in another one, syntactically similar documents and different ones. In order to detect if two documents are syntactically similar or share the common part the additional word-by-word document comparison should be performed.

During the testing phase the method, based on the modified private-matching scheme, showed that it is very slow and isn't suitable for the privacy-preserving document similarity detection. Another method, on the contrary, was found to have very good performance and can be used for creation of the tool for privacy-preserving document detection. But it needs some improvements, such as adding the extra comparison of all documents words when it might be required. More statistics should be accumulated to assure that the algorithm detects the document similarity types in a proper way.

References

[1] G. Salton et al. A vector space model for automatic indexing. Communications of the ACM 18(11), Nov. 1975

[2] K.M. Hammouda, M. S. Kamel. Document similarity using a phrase indexing graph model. Knowledge and Information Systems 6(6), Nov. 2004

[3] N. Shivakumar, H. Garcia-Molina. SCAM: a copy detection mechanism for digital documents. In proceedings of the 2nd International Conference in Theory and Practice of Digital Libraries (DL'95), Austin, Texas, USA, Jun. 11-13 1995

[4] E. Garsia. Cosine similarity and Term Weight Tutorial: an information retrieval tutorial on cosine similarity measures, dot products and term weight calculations. http://www.miislita.com/information-retrieval-tutorial/cosine-similarity-tutorial.html, 2006

[5] H. Pang, J. Shen, R. Krishnan. Privacy-preserving similarity-based text retrieval. ACM Transactions on Internet Technology 10(1), Feb. 2010

[6] M. Murugesan et al. Efficient privacy-preserving similar document detection. The VLDB Journal — The International Journal on Very Large Data Bases 19(4), Aug. 2010

[7] Wei Jiang et al. Similar document detection with limited information disclosure. In proceedings of IEEE 24th International Conference on Data Engineering, Cancun, Mexico. Apr. 7-12 2008

[8] J. Vaidya, C. Clifton. Privacy preserving association rule mining in vertically partitioned data. Sixth IEEE International Conference on Data Mining (ICDM06), Hong Kong, China, pp. 1070-1075, Dec 18-12 2006

[9] M. O. Rabin. Fingerprinting by random polynomials. Center for Research in Computing Technology, Harvard University, Tech Report TR-15-81, 1981

[10] A. Z. Broder. On the resemblance and containment of documents. In proceedings of the Compression and Complexity of Sequences 1997. Salerno, Italy, Jun 11-13 1997

[11] M. Malkin, R. Venkatesan. Comparison of texts streams in the presence of mild adversaries. ACSW Frontiers '05 Proceedings of the 2005 Australasian workshop on Grid computing and e-research - Volume 44.

[12] Udi Manber. Finding similar files in a large file system. WTEC'94: Proceedings of the USENIX Winter 1994 Technical Conference on USENIX Winter 1994 Technical Conference. Jan. 1994

[13] Daniel Micol, Oscar Ferrandez et al. A textual-based similarity approach for efficient and scalable external Plagiarism Analysis. Lab Report for PAN at CLEF2010 (Conference on Multilingual and Multimodal Information Access Evaluation), Padua, Italy, Sep 20-23 2010

[14] M. J. Freedman et al. Efficient private matching and set intersection. In proceedings of International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2004), Interlaken, Switzerland, May 2-6 2004

[15] V.A. Narayana, P.Premchand, A.Govardhan. Fixing the threshold for effective detection of near duplicate web documents in web crawling. In proc.: ADMA'10 Proceedings of the 6th international conference on Advanced data mining and applications: Part I. Chongqing, China, Nov 19-21 2010

[16] Mohamed Elhadi and Amjd Al-Tobi. Part of speech(POS) tags sets reduction and analysis using rough set techniques. RSFDGrC '09 Proceedings of the 12th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing. Delhi, India, Dec. 15-18 2009

[17] I. Spasic, S. Ananiadou et al. Text mining and ontologies in biomedicine: making sense of raw text. Brief Bioinform 6 (3) 239-251, 2005.

[18] V. Oleshchuk, A. Pedersen. Ontology Based Semantic Similarity Comparison of Documents. DEXA '03 Proceedings of the 14th International Workshop on Database and Expert Systems Applications. Prague, Czech Republic, Sep. 1-5 2003

[19] A. Stavrianou, P. Andritsos, N. Nicoloyannis. Overview and Semantic Issues of text mining. ACM SIGMOD Record 36(3), Sep. 2007

[20] Liddy, E.D. 2001. Natural Language Processing. In Encyclopedia of Library and Information Science, 2nd Ed. NY. Marcel Decker, Inc.

[21] C. D. Manning, P. Raghavan, H. Schütze. Introduction to Information Retrieval. Cambridge University Press, 2008.

[22] P. Chen et al. A fully unsupervised word sense disambiguation method using dependency knowledge. NAACL '09 Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Stroudsburg, USA, May 21 –Jun 5 2009

[23] http://www.intellexer.com/solutions.html

[24] Sentiment analysis and language processing tools, http://lordpimpington.com/codespeaks/drupal-5.1/?q=node/5

[25] http://www.cogilex.com/products.htm

[26] X.R. Hu, E Atwell. A survey of machine learning approaches to analysis of large corpora. Proceedings of the Workshop on Shallow Processing of Large Corpora, Lancaster University, UK, pp. 45–52, Mar 26 2003.

[27] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. Advances in Cryptology – EUROCRYPT'99, Prague, Czech Republic, May 2-6 1999

[28] J. Massey, J. Omura. Method and apparatus for maintaining the privacy of digital message conveyed by public transmission. United States Patent number: 4567600

[29] http://www.nltk.org/

[30] Steven Bird. Natural Language Processing with Python. O'Reilly, 2009

[31] http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

[32] http://nlp.cs.nyu.edu/GMA_files/resources/english.stoplist

[33] http://www.ranks.nl/resources/stopwords.html

[34] http://en.wikipedia.org/wiki/List_of_English_prepositions

[35] http://www.englishclub.com/vocabulary/prepositions.htm

[36] http://www.advanced-english-grammar.com/preposition-list.html

[37] http://www.paulnoll.com/Books/Clear-English/English-adverbs.html

[38] http://www.esldesk.com/vocabulary/adverbs/list-3

[39] http://www.momswhothink.com/reading/list-of-adverbs.html

[40] http://en.wikipedia.org/wiki/Natural_language_processing_toolkits

[41] http://lordpimpington.com/codespeaks/drupal-5.1/?q=node/5

[42] http://pypi.python.org/pypi/stemming/1.0

[43] A.Kim, J. Luo, M. Kang. Security ontology for annotating resources. Naval Research lab Washington DC, Memorandum report: NRL/MR/5542-05-8903, Aug 31 2005.

[44] http://www.ai.sri.com/daml/services/owl-s/security.html

[45] http://www.ida.liu.se/~iislab/projects/secont/

[46] C. Blanco et al. A systematic review and comparison of security ontologies. ARES '08 Proceedings of the 2008 Third International Conference on Availability, Reliability and Security. Washington DC, USA, 2008.

[47] RFC2828

[48] M. Stamp. Information security: Principles and practice. John Wiley & Sons, Inc. 2006.

[49] W. Stallings, L. Brown. Computer Security: Principles and Practice. Prentice Hall, 2008.

[50] http://portal.acm.org/dl.cfm

[51] http://ieeexplore.ieee.org/Xplore/guesthome.jsp

Appendix A

PoS database files contents

Stopwords_mult.txt

a's	isn't	weren't	mayn't
ain't	it'd	what's	mightn't
aren't	it'll	where's	she'll
c'mon	it's	who's	she's
c's	let's	won't	that'll
can't	shouldn't	wouldn't	that've
couldn't	t's	you'd	there'd
didn't	that's	you'll	there'll
doesn't	there's	you're	there're
don't	they'd	you've	there've
hadn't	they'll	no one	what'll
hasn't	they're	now nowadays	what've
haven't	they've	out outside	who'd
he's	thoroughly	he'd	who'll
here's	wasn't	he'll	how's
i'd	we'd	daren't	when's
i'll	we'll	co.	why's
i'm	we're	e.g.	et.al.
i've	we've	inc.	proc.

Stopwords_one.txt

a	got	others	whereafter	dost	srd
ab	gotten	ought	whereas	doth	stave
able	greetings	our	whereby	double	staves
according	h	ours	wherein	dr	studies
allow	had	ourselves	whereupon	dual	supposing
allows	happens	overall	wherever	due	ta
although	has	own	whether	eb	tested
am	have	p	which	ec	thee
an	having	particular	while	eu	thenceforth
and	he	placed	whither	excepted	thereabout
another	hello	please	thereupon	exception	thereabouts
anybody	help	possible	they	exclude	thereof
anyhow	her	provides	think	exclusive	thereon
anyone	hereafter	q	third	fa	thereto
anything	hereby	que	thorough	fae	tho
anyways	herein	qv	those	farther	thou
appear	hereupon	r	though	farthest	thy
appreciate	hers	rd	three	ff	thyself
appropriate	herself	re	thru	found	time
are	hi	regards	thus	free	types
ask	him	s	took	front	unable
asking	himself	said	tried	furthest	upwards
associated	his	same	tries	gcn	vol

available	hither	saw	try	general	week
b	howbeit	say	trying	gif	whatsoever
be	i	saying	two	gm	whensoever
became	ie	says	u	halves	whereabouts
become	if	second	un	hast	whereat
becomes	ignored	see	unless	hath	wherefore
becoming	immediate	seeing	unto	henceforth	wherefrom
been	inasmuch	seem	us	hereabouts	whereinto
being	inc	seemed	use	hereto	whereof
believe	indeed	seeming	used	hindmost	whereon
both	indicate	seems	useful	hitherto	wheresoever
brief	indicated	seen	uses	howsoever	whereto
c	indicates	self	using	hr	whereunto
came	inner	selves	uucp	include	wherewith
can	insofar	sensible	v	included	whew
cannot	is	sent	value	indoors	whichever
cant	it	serious	whole	insomuch	whichsoever
cause	its	seven	whose	investigated	whilst
causes	itself	several	will	jpg	whoa
certain	j	shall	willing	kg	whomever
changes	k	she	wish	kind	whomsoever
co	keep	should	wonder	km	whosoever
com	keeps	six	would	la	wilt
come	kept	some	x	low	wow
comes	know	somebody	y	made	www
consider	knows	someone	you	meantime	ye
contain	known	something	your	mr	year
containing	l	sometime	yours	mrs	yippee
contains	latter	sorry	yourself	ms	ble
corresponding	lest	specified	yourselves	net	adj
could	let	specify	z	news	begin
course	liked	specifying	zero	nope	caption
d	look	sub	item	nu	dare
described	looking	such	login	obtained	eighty
did	looks	sup	ac	org	end
different	ltd	sure	ads	page	ending
do	m	t	ae	performance	fewer
does	may	take	af	performed	half
doing	me	taken	albeit	pl	hundred
done	mean	tell	asp	plenty	make
downwards	might	tends	author	post	makes
e	moreover	th	av	present	mine
each	must	thank	baf	presented	miss
edu	my	thanks	bf	presents	neverf
eg	myself	thanx	biz	provide	neverless
eight	n	thats	ca	provided	ninety
either	name	the	canst	related	recent
else	nd	their	cd	report	someday
et	necessary	theirs	cee	required	taking
etc	need	them	cf	results	thing
everybody	needs	themselves	cfm	roll	things
everyone	neither	thence	cfrd	sake	thirty
everything	nevertheless	thereafter	cgi	sang	undoing

ex	new	thereby	choose	save	al
example	nine	therefore	click	selected	nt
f	nobody	therein	cm	sfrd	md
few	non	theres	conducted	shalt	pp
fifth	none	various	considered	show	1st
five	noone	viz	contrariwise	shows	2nd
followed	nor	vs	cos	shown	3d
follows	not	w	crd	significant	3rd
former	nothing	want	cu	slept	4th
four	novel	wants	cx	slew	5th
g	o	was	date	slung	6th
get	oh	way	day	slunk	7th
gets	ok	we	describes	smote	8th
getting	okay	welcome	designed	sort	9th
gives	old	went	determine	spat	10th
go	one	were	determined	spoke	0
goes	ones	whatever	dfe	spoken	ps
going	or	whence	discussed	sprang	pc
gone	other	whenever	dont	sprung	ca

Preposit_mult.txt

in time	outside of	in return	to the detriment of
in the forefront of	once in a while	in reverse	to the exclusion of
at the forefront of	of course	in short	to the full
in demand	side by side	in succession	under cover of
on demand	side to side	in terms of	under lock and key
in focus	so that	in the aftermath	with a view to
out of focus	very much	in the balance	with an eye to
in touch	during long time	in the case of	with regard to
out of touch	during short time	in the course of	with respect to
in sight	for lack of	in the event of	with the exception of
within sight	for life	in the extreme	as far as
in a flash	for love	in the eyes of	ahead of
in a hurry	for real	in the flesh	according to
in a mess	for the good of	in the form of	along with
in a sense	for the sake of	in the habit of	as per
in advance	for want of	in the light of	as regards
in agreement with	in accordance with	in the long run	aside from
in aid of	in addition to	in the meantime	as well as
in all likelihood	in case of	in the name of	at the outset
in an instant	inside of	in the open	at the end
in brief	instead of	in the space of	at sight
in bulk	in front of	in the wake of	at the double
in common	in lieu of	in the way of	at a time
in comparison with	in place of	in theory	at a glance
in confinement	in point of	in time for	at a loss
in conjunction with	in spite of	in times of	at a low ebb
in connection with	in the end	in tune with	at a moment's notice
in consequence of	owing to	in turn	at all costs
in contrast with	on behalf of	in two minds	at all events
in contrast to	on sight	in unison	at any cost
in disorder	on the double	in vain	at any rate

in due course	prior to	in view of	at ease
in earnest	pursuant to	in words	at large
in effect	regardless of	behind schedule	at least
in essence	thanks to	on the air	at length
in excess of	that of	of the air	at most
in exchange for	to the benefit	on balance	at once
in fact	where as	off balance	at one time
in favor of	with respect to	on a regular basis	at present
in full	a bit	on account of	at random
in gear	a lot	on average	at the beginning
in general	how many	on condition that	at the expense of
in good faith	how much	on display	at the foot of
in hand	how long	on the brink of	at the hands of
in harmony with	how tall	on the dot	at the height of
in harmony	how high	on the edge of	at the latest
in haste	hardly ever	on the eve of	at the mercy of
in hiding	next to	on the grounds of	at the peak of
in line with	in operation	on the horizon	at the same time
in mind	in opposition to	on the hour	at the top of
in moderation	in other words	on the off-chance	at this juncture
in name	in particular	on the part of	at times
in spare time	in person	on the point of	because of
by virtue of	in pieces	on the strength of	by means of
by way of	in practice	on the stroke of	by chance
behind the scenes	in preference to	on the way to	by any chance
close to	in principle	on time	by the name of
except for	in private	on tiptoe	by luck
far from	in proportion to	out of print	by accident
far away	in proportion with	in print	by all accounts
from the outset	in public	out of step	by all means
for a change	in pursuit of	in step	by dint of
for certain	in quantity	out of date	by far
for sure	in reality	out of hand	by force
for fear of	in recognition of	out of the question	by hand
for good	in relation to	to the contrary	by heart
for granted	in reply to	on the contrary	by no means
for hire	in respect of	to an extent	by oneself
over there	in response to	to date	by sight
on top of	in retrospect	to excess	by the side of

Preposit_one.txt

abaft	around	beyond	given	over	to
aboard	aside	but	hence	on	toward
about	astride	because	into	out	towards
above	athwart	by	including	past	underneath
abroad	atop	behind	in	per	unlike
absent	ago	concerning	inside	plus	until
across	apart	considering	like	pace	up
afore	away	circa	minus	pro	upon
after	as	despite	mid	qua	under
against	at	down	midst	regarding	versus
along	before	during	near	round	via

alongside	barring	except	next	since	vice
amid	below	excepting	notwithstanding	sans	within
amidst	beneath	excluding	of	than	worth
among	beside	following	off	through	withal
amongst	besides	failing	onto	throughout	with
anti	between	for	opposite	till	without
apropos	betwixt	from	outside	times	

Adverb_mult.txt

arm-in-arm	forward-and-back	non-stop	right-to-left
back-and-forth	hand-in-hand	now-and-then	side-by-side
back-to-back	head-first	now and then	side-to-side
both-ways	head-to-toe	off-key	to-and-fro
counter-clockwise	just-so	off-tune	up-and-down
face-first	left-and-right	right-and-left	upside-down
feet-first	left-to-right	rightside-up	right now

Adverb_one.txt

afterwards	better	headlong	not	seldom	these
ahead	best	higher	now	sometimes	upright
almost	clearer	however	nowhere	somewhat	unannounced
aloud	clockwise	here	nearer	somewhere	unawares
already	closer	home	nonetheless	soon	underfoot
also	close	how	northward	sooner	unseen
altogether	deadpan	instead	nearby	soonest	upbeat
always	deeper	inward	no	sideways	upward
anew	downward	inwards	often	skyward	underground
anymore	downstairs	just	once	slower	upstairs
anywhere	downtown	left	onward	soaked	very
askew	doubtless	less	otherwise	softer	verbatim
aslant	earlier	ladylike	outward	somehow	well
awry	even	leftward	outwards	southward	westward
again	evermore	lengthwise	overhead	still	withdrawn
agape	eastward	likewise	overmuch	straight	worrisome
alone	elsewhere	lots	paler	stupefied	worse
amuck	everywhere	louder	parallel	so	worst
anyway	every	lower	partway	then	why
askance	ever	last	pointblank	tomorrow	when
awhile	enough	later	pretty	too	where
aback	far	little	perhaps	thrice	who
afresh	fast	least	quicker	tighter	what
all	forever	more	quite	together	whoever
any	forth	meanwhile	rearward	twice	whom
back	forward	much	right	there	yesterday
backward	further	most	red	today	yet
backwards	furthermore	maybe	regardless	tonight	yeah
beforehand	hard	many	rightward	this	yes
bent	harder	never	rather	that	

Non_adverb_ly.txt

sully bully rely ally	rally fly butterfly ply	sly apply italy

Appendix B

Ontology database files content

Ontology_mult.txt

enforcement of the security policy:authorization	simple security property:bell-lapadula
access control center:authorization	confidentiality service:authorization
access control service:authorization	confidentiality services:authorization
access control services:authorization	confidential policy:authorization
access control mechanism:authorization	data privacy:authorization
protection of data:authorization	access permission:authorization
access control mechanisms:authorization	access permissions:authorization
privilege management infrastructure:authorization	access privilege:authorization
role based access control:role-based access control	access privileges:authorization
dynamic separation of duty:role-based access control	authorized entity:authorization
static separation of duty:role-based access control	authorized entities:authorization
group of users:role-based access control	authorization process:authorization
groups of users:role-based access control	security clearance:authorization
role-based access control:role-based access control	security policy:authorization
separation of duty:role-based access control	security service:authorization
rule-based security policy:role-based access control	security services:authorization
discretionary access control:discretionary access control	secure state:authorization
identity-based access control:discretionary access control	security violation:authorization
access control list:discretionary access control	sensitive information:authorization
delegation of rights:discretionary access control	sensitive resources:authorization
access control matrix:discretionary access control	sensitive resource:authorization
mandatory access control:mandatory access control	unauthorized access:authorization
rule-based access control:mandatory access control	unauthorized manner:authorization
policy of competence:mandatory access control	unauthorized disclosure:authorization
military security policy:mandatory access control	unauthorized use:authorization
chinese wall policy:mandatory access control	system resources:authorization
clark-wilson integrity model:mandatory access control	system resource:authorization
clark wilson model:mandatory access control	system entity:authorization
multilevel-secure computer system:multilevel security	limit access:authorization
multi-level access control:multilevel security	data security:authorization
multilevel access control:multilevel security	restrict access:authorization
multilevel security mode:multilevel security	data protection:authorization
security clearance level:bell-lapadula	protected status:authorization
security classification level:bell-lapadula	access control:authorization
security clearance levels:bell-lapadula	access mode:authorization
security classification levels:bell-lapadula	privilege process:authorization
rbac model:role-based access control	access modes:authorization
access right:role-based access control	access right:authorization
access rights:role-based access control	access rights:authorization
assigned role:role-based access control	multilevel security:authorization
assigned roles:role-based access control	multilateral security:authorization
core rbac:role-based access control	dedicated security mode:authorization
constrained rbac:role-based access control	eligible to access:authorization
consolidated rbac:role-based access control	denial of service:authorization
hierarchical rbac:role-based access control	mandatory access control:authorization
limited hierarchy:role-based access control	role-based access control:authorization
limited hierarchies:role-based access control	role based access control:authorization
general hierarchy:role-based access control	discretionary access control:authorization
general hierarchies:role-based access control	identity-based access control:authorization
least privilege:role-based access control	rule-based access control:authorization

least privileges:role-based access control permission assignment:role-based access control user assignment:role-based access control biba model:mandatory access control biba's model:mandatory access control clark-wilson model:mandatory access control classified status:mandatory access control classification category:mandatory access control security labels:mandatory access control security label:mandatory access control clearance level:mandatory access control clearance levels:mandatory access control security clearance:mandatory access control classification level:mandatory access control classification levels:mandatory access control need-to-know model:mandatory access control access matrix:discretionary access control capability ticket:discretionary access control capability list:discretionary access control capability lists:discretionary access control dac model:discretionary access control need-to-know:mandatory access control attribute-based access control:authorization	star-property:bell-lapadula ss-property:bell-lapadula ds-property:bell-lapadula bell-lapadula:bell-lapadula *-property:bell-lapadula top secret:multilevel security secure state:bell-lapadula confinement property:bell-lapadula simple-security condition:bell-lapadula security clearance:bell-lapadula top secret:bell-lapadula security classification:bell-lapadula security class:bell-lapadula tranquility property:bell-lapadula bell-lapadula:multilevel security need-to-know:multilevel security multilevel security:multilevel security multi-level security:multilevel security security level:multilevel security security clearance:multilevel security need-to-know model:multilevel security confidentiality policy:multilevel security

Ontology_one.txt

assign:role-based access control assigned:role-based access control constrain:role-based access control constrains:role-based access control hierarchy:role-based access control hierarchies:role-based access control operation:role-based access control operations:role-based access control permission:role-based access control permissions:role-based access control privilege:role-based access control privileges:role-based access control right:role-based access control rights:role-based access control classification:multilevel security secret:multilevel security confidential:multilevel security unclassified:multilevel security clearance:multilevel security dac:discretionary access control classification:mandatory access control classified:mandatory access control clearance:mandatory access control unclassified:mandatory access control mac:mandatory access control authorization:authorization authorize:authorization authorized:authorization authorizes:authorization	grants:authorization identity:authorization mac:authorization object:authorization objects:authorization permission:authorization permissions:authorization permitted:authorization permit:authorization permits:authorization privilege:authorization privileges:authorization resources:authorization resource:authorization rbac:authorization subject:authorization subjects:authorization system:authorization user:authorization violation:authorization violate:authorization violated:authorization violates:authorization unauthorized:authorization confidentiality:bell-lapadula clearance:bell-lapadula secret:bell-lapadula confidential:bell-lapadula unclassified:bell-lapadula

authorizing:authorization	classification:bell-lapadula
confidentiality:authorization	mls:authorization
dac:authorization	disclosed:authorization
disclosure:authorization	disclose:authorization

Ontology_structure.txt

authorization:authorization:89:access control
role-based access control:authorization:39:rbac,role based access control
discretionary access control:authorization:11:dac
mandatory access control:authorization:25:mac
multilevel security:authorization:18:mls
bell-lapadula:multilevel security:24: