Privacy-Preserving Text Mining as a Service

Gianpiero Costantino, Antonio La Marra, Fabio Martinelli, Andrea Saracino, Mina Sheikhalishahi
Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Pisa, Italy
Email: name.surname@iit.cnr.it

***Abstract*—Text mining is the process to automatically infer relevant information from semantically related text documents. This technique, which has applications from business intelligence to homeland security, terrorism and crime fight, might bring noticeable privacy issues when analyzed documents contain privacy sensitive information. In this paper, we propose a framework for privacy-preserving text analysis, which exploits Homomorphic Encryption, to analyze text documents in a privacy preserving manner. The proposed framework is designed to ensure that there is no disclosure of privacy sensitive information contained in the document to any party, including the analysis engine itself. Furthermore, we present two use cases of analysis based on bag-of-words classification, where the proposed framework manages to obtain good classification results without information disclosure. In particular the two different settings that are considered are: tweet analysis for detection of terrorist Twitter accounts, and out-box mail analysis for detection of bot infected devices. Accuracy results with different classifiers, performances and a security analysis of our approach are presented and discussed.**

I. Introduction

Nowadays, on-line providers offers remote services, e.g., data-storage on Cloud, that are convenient to their customers. These services are mostly provided by companies like Dropbox, Google, iCloud, OneDrive and others. On one side, public providers are very usable, scalable and allow users to access several interesting services for free. On the other side, public providers are seen as critical point of failure when thinking at the privacy of users' data. Information stored in these systems should be considered private, since belonging to different entities, generally embodied by physical users. However, these information are often object of analysis, since provide additional knowledge about users and service usage, which are exploited either commercially, or to support analysis for law authorities [20] [9]. Performing data analysis without violating the privacy, i.e. Privacy Preserving Data Analysis, is a challenging task, requiring the use of advanced encryption techniques and the design of specific algorithms to ensure the trade-off between analysis result accuracy and ensured privacy [15] [14].

A technique which ensures both high accuracy and elevated privacy is the Homomorphic Encryption (HE) [2], which is an encryption schema to perform computations directly on cipher-texts, such that the encrypted result of a HE function executed on encrypted data returns the same result of the same function using clear-texts. However, this approach is rarely used due to the massive performance overhead, and the general low expressiveness of operations which can be performed homomorphically.

In this paper, we present a solution to recognize and classify text-based data, offloaded into the Cloud or, more generally, to remote servers, by preserving the privacy of their content exploiting HE. We introduce a framework model that can be adapted from a service provider where an *HE service provider* offers HE operations according to the *Software as a Service* (SaaS) model, to entities interested in analyzing data (*data Analyzers*) on information stored by *data Providers*. The presented framework allows thus, data analyzers to obtain accurate data analysis results without disclosing the information of the data providers neither to the analysis nor to the HE service provider. Hence, two real use cases are presented whose information is privacy sensitive and their analysis is of public interest. In particular, we will discuss (i) an application of privacy-preserving text analysis on Twitter posts (*Tweets*) to discover terrorist support messages and accounts, and (ii) the application of privacy preserving text analysis of email out-boxes to find accounts belonging to spam botnets. The data analysis technique discussed in this work is based on supervised machine learning and the *bag-of-words* approach for text classification [25]. Afterward, classification analysis results will be presented, comparing the results of different classifiers, where an accuracy higher than 94% is reached for both the proposed classification problems.

The privacy-preserving services are developed using the HElib[1] software library. HElib provides libraries that can be used coding in "C" to write functions that will be executed using the homomorphic encryption. We wrote our services by integrating HElib to evaluate the possible terrorism messages and spam emails. Services are exposed at the server-side, and the results of our experiments show that although our implementation preserves privacy of data analyzed, the computational-time needed to process small sets of data does not represent an issue. Instead, an opposite observation can be pointed out when the set dimension is large.

Section II provides details on text-analysis techniques and gives a background on homomorphic encryption. Section III describes the architecture of the proposed framework for text-analysis in privacy-preserving fashion and the threat model. In Section IV, we present the application of the proposed framework on two use cases, we show the performances of our approach and we provide a security analysis of it. Section V presents related works to the presented one and Section VI briefly concludes the paper and proposes some future directions.

[1] https://github.com/shaih/HElib

II. Background

In this section we will review the concept related to text analysis and text classification, followed by some notions on homomorphic encryption.

A. Text Analysis

Text categorization (or *text classification*) is the assignment of texts to predefined categories according to their content. Automatic text categorization has many practical applications , including indexing for document retrieval [19], catalog news articles [12], automatically extracting metadata, automatically learning of reading interests of users [21], automatic sorting of electronic mails, word sense disambiguation by detecting the topics a document covers, organizing and maintaining large catalogs of Web resources [3], and in general any application that requires document organization or selective and adaptive document dispatching [25]. Formally, text categorization is the task of assigning a *Boolean* value to each pair $< t_i, c_j > \in \mathcal{T} \times \mathcal{C}$, where $\mathcal{T}$ is a domain of texts and $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$ is a set of predefined categories. A value *True* assigned to $< t_i, c_j >$ indicates that the document (text) t_i belongs to the category c_j, and consequently the value *False* means that t_i does not belong to the category c_j. The task of finding the category of each text is generally executed through a *classifier* that can be defined as a function $\bar{\Phi} : \mathcal{T} \times \mathcal{C} \rightarrow \{True, False\}$ that approximates an unknown target function $\Phi : \mathcal{T} \times \mathcal{C} \rightarrow \{True, False\}$ [7].
Typical approaches for text categorization extract feature from each document, and use the feature vectors as input to model a classifier [17] [25]. With the use of words as features, probably a small number of well-chosen words and word occurrence counts as feature values, a *classifier* is constructed for a set of documents. The documents, which belong to a specific category, are positive samples (*True*) and the remaining documents negative ones (*False*). The *classifier*, generally by considering the highest likelihood, predicts whether or not that category is assigned to a new document based on the words in it, and their occurrence counts. These text categorization models that utilize *words* and their frequency as the features for training a classifier, is called *bag-of-words* model [11]. In this work, we exploit the *bag-of-words* model to build a classifier on the applied documents.

B. Homomorphic Encryption

Homomorphic Encryption (HE) is an encryption method to perform computation on encrypted data without decrypting it. Let's suppose that *Alice* is a customer of the service provider *Bob*. He allows Alice to offload her data in a remote disk space, e.g., Cloud, to retrieve her files in any moment, anywhere. Alice does not know Bob and she does not trust in Bob and his way to manage her files. So, Alice decides to encrypts her files before offloading them into the remote disk. Sometimes, Alice needs to search through her files to download the files that she needs. However, since the files are encrypted she needs to download the file, decrypt and check if it is the file she needs. Now, the search operation is very complicated to be performed, instead Alice requires a method to search among encrypted files.

The HE will be the solution that Alice needs to protect her files from Bob. So, Alice needs a scheme that has homomorphic properties, such as:

$$\begin{array}{lcl} c_1 & = & Enc(m_1) \\ c_2 & = & Enc(m_2) \\ m_1 + m_2 & = & Dec(c_1 + c_2) \end{array} \quad (1)$$

The schema represented in 1 is homomorphic for addition. A schema that is homomorphic for addition and multiplication is called fully homomorphic, i.e., Fully Homomorphic Encryption (FHE). The breakthrough work on FHE was presented by Gentry [8] in 2009 in which he showed how to compute arbitrary functions on encrypted data, but computational and data storage overheads were very high. The first schema proposed by Gentry supported only a bounded number of additions and multiplications, and this is why it is considered "Somewhat Homomorphic"(SH). Afterwards, he presented a way to turn from SH to FH using a technique that he calls *bootstrapping*, or refreshing, in which the decryption procedure is expressed with a low polynomial degree in the bits of both ciphertext and secret keys. In 2012, Gentry with Brakerski and Vaikuntanathan (BGV) [2] improve his schema without bootstrapping by suggesting a generalized secure construction under the popular Learning With Errors assumption and its ring variant. In 2012, Fan and Vercauteren (FV) [6] proposed a ring variant scheme and improved its efficiency. The software library HElib [10] is an implementation of the Brakerski-Gentry-Vaikuntanathan (BGV) scheme that allows researcher to work with homomorphic encryption (HE) using low-level routines such as: addition, multiplication, shifting and others. Due to its functional properties, we use HElib library in our framework for the Homomorphic Encryption evaluation.

III. Architecture and Workflow

In this section, we will describe the architecture of the proposed framework and the envisioned workflow, first in a general form and then applied to text mining for word frequency analysis.

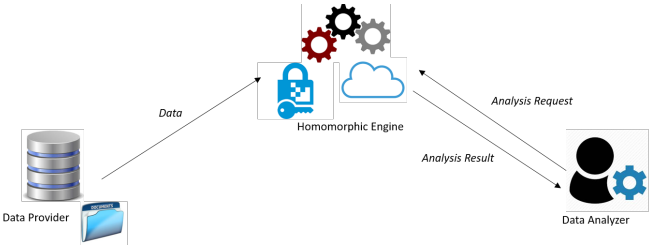

Fig. 1: Architecture of the proposed framework.

The envisioned system for homomorphic data analysis consists of three main actors, depicted in Figure 1. The *Data Provider* collects and stores records in a text format, which contain privacy sensitive information, that should not be accessed without authorization. The *Homomorphic Engine*

is a component that performs privacy preserving data analysis operations. Finally, the *Data Analyzer* is the component that performs data mining and machine learning techniques for pattern recognition, market basket analysis, keyword matching and other operations on the results of the homomorphic engine.

A. Workflow and Application

The Homomorphic Engine (HEng) receives data by both the Data Provider (DP) and the Data Analyzer (DA) in encrypted form. Data processed by the HEng are encrypted with the DA's public key. Hence, source data cannot be read, given that the HEng follows the protocol steps, and afterwards, the DA receives the result in an encrypted manner. Finally, the result is decrypted and processed for further analysis.

Though the model can be generalized to different kind of analysis, this work is focused on text analysis, where the Data Analyzer wishes to find the number of occurrences of specific words in privacy sensitive texts stored by the Data Provider. Words occurrences are then used to perform bag-of-words classification by exploiting machine learning classifiers. Let's suppose the DA owns a set of m words whose occurrences have to be found inside the analyzed text. This set of words is fixed for any specific analysis and will be represented in the following as x_w. The words occurrences has to be found in one or more documents x_d stored by the data provider. Due to their computational complexity, operations which can be performed through homomorphic encryption are quite simple and generally take integer numbers as parameter. Hence, it is needed to translate the operation of word matching in terms of algebraic operations among integer number. This task, can be performed by hashing (through SHA-1) the words in x_w and doing the same for each word in x_d, as shown in Figure 2. The words in x_w are thus hashed, to be represented as

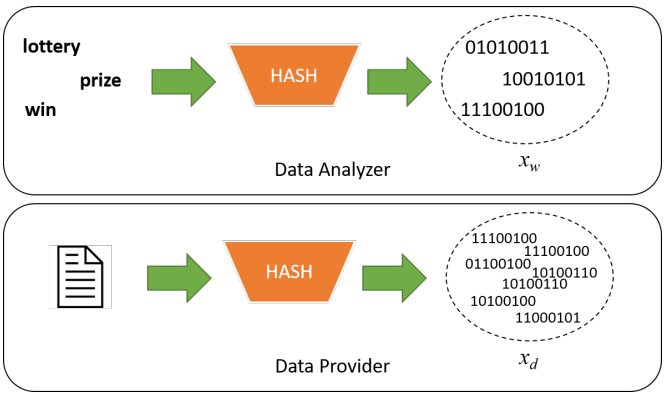

Fig. 2: Text and word hashing by provider and analyzer.

integer numbers, as well as the ones contained in x_d, which are also scrambled after hashing, to further complicate the task of inferring the text structure. The Homomorphic Engine works on the hashed data, finding an occurrence of a specific word x_{w_i} in x_d if for a specific word $x_{d_j} \in x_d$ it is verified that $h(x_{d_j}) - h(x_{w_i}) = 0$, where $h()$ represents the hash function. This operation can be easily translated in an homomorphic subtraction, where the effective value of all the words in x_d is never disclosed, as depicted in Figure 3. As shown, the final

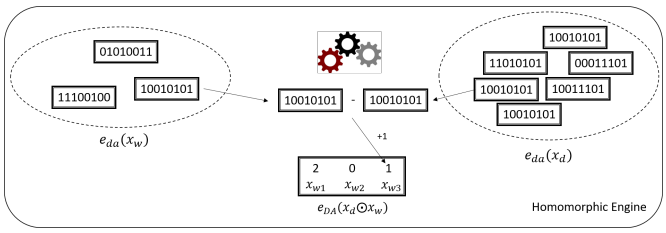

Fig. 3: Homomorphic subtraction and vector generation.

result of the operation is a vector of m elements containing the occurrences of each word in x_w found in x_d. Finally, the result of this operation is returned still encrypted to the DA, which can feed the vector to a pre-trained classifier to effectively perform the bag-of-words classification, as shown in Figure 4.

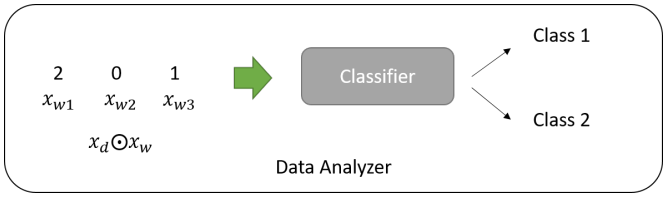

Fig. 4: Homomorphic subtraction and vector generation.

Figure 5 shows the whole work flow of the proposed framework and it is detailed as following.

1) The Data Analyzer shares with the Homomorphic Engine and the Data Provider its public key e_{DA}.
2) The Data Provider encrypts texts x_d with e_{DA} and sends them to the Homomorphic Engine.
3) The Data Analyzer sends the encrypted set of words of which it wants to discover their frequencies x_w.
4) The Homomorphic Engine finds the number of occurrences of words in x_w, inside the text x_d, without decrypting them. Hence, HEng returns the encrypted result to the DA, where the $\odot$ symbol, represents the homomorphic operation.

After decryption, the received result, corresponds to a vector of m numbers, representing the number of occurrences of each word in the text x_d. The decrypted numerical vectors are hence classified by a pre-trained classifier that identifies the belonging class of a message depending on the frequency of specific words.

B. Threat Model

To study the soundness of our framework against different attacks, we start from a wide observation that considers threats coming from both inside and outside the framework.

The insider *(internal)* threats focus on the behaviors of the Data Analyzer and Homomorphic encryption Engine, which might try to read and eventually maliciously re-use any information that can be leaked from the Data Provider. In this

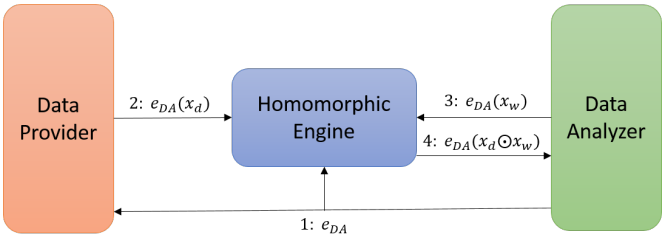

Fig. 5: Homomorphic Analysis Protocol.

hypothesis, an attack from inside the framework will succeed if: (i) the Homomorphic encryption Engine is able to read all or part of the data stored by the Data Provider, or is able to read all or part of the operands sent by the Data Analyzer; (ii) the Data Analyzer is able to read or infer part or all of records stored by the Data Provider.

In particular, we envision the following attacks:

- *Known cypher-text* attack: here, the Data Analyzer or the Homomorphic encryption Engine tries to extract the content of an encrypted message.
- *Dictionary* attack: the attacker selects a set x_w of words, and is able to receive or compute the corresponding cypher-text, still cannot decrypt it. The attacker, for example the homomorphic encryption engine, thus, attempts to build a dictionary cypher-text/plain-text, to infer interesting words directly from the cypher-text.
- *Record leakage* attack: the attacker, such as the Data Analyzer, properly forges x_w as a dataset of words useful to identify records related to a specific entity, for instance a physical person.

Attackers *outside* the framework will likely attempt to perform a Man-in-the-middle (MITM) attack to read data in the communication between the Data Provider and the Homomorphic Engine. In this case, these attacks can be led back to the*known cyphertext* attacks. Finally, impersonification attacks and malicious data analyzers will not be considered in this work. The aforementioned attacks will be addressed in Section IV-E.

IV. Analysis and Results

In this section we present the application of the proposed framework on two use cases, where word frequency and bag of word classification are used to identify critical accounts in social networks and email services. Afterward, classification results will be presented to demonstrate the effectiveness of the proposed approach.

A. Terrorism Support Tweet Analysis

Twitter is one of the most popular micro bloging on-line social network today [4]. It lets the users to share short messages, named *tweets* with their "followers". The popularity of Twitter makes it a perfect mean for reaching large numbers of people, thus has been largely used in last years also for spreading political and religious ideas, support solidarity campaigns and more. Based on its enormous pervasiveness, Twitter has clearly succeeded in its main goal of providing a ubiquitous real-time push-based information sharing platform to everyone [4].

Unfortunately, Twitter has also been used recently, as a mean to distribute terrorist propaganda, with posts coming from several accounts, supporting and praising terror attacks recently happened all around the world. ISIS (Islamic State in Iraq and Syria), which is currently the strongest organization of terrorists, abused this platform to distribute their propaganda in order to attract supporters, claim responsibility of attacks and threaten European countries and US. The accounts supporting ISIS are not blocked or banned by purpose, since the tweet analysis gives useful information exploited by authority to prosecute the persons behind the account. Therefore, the early detection of terrorist tweets is vital and such a differentiation can be done through the *bag-of-words* approach [13]. Performing such an analysis through the methodology proposed in this work, it would allow to analyze tweets regardless of the privacy settings of the specific account. In fact, since there is no risk of disclosure, it is possible to consider Twitter itself as a data provider, which will provide tweets, without disclosing neither the effective text nor the account owner. After the analysis has been performed, suspicious tweets can be separated and further analyzed, minimizing thus the risk of violating the privacy of not involved users.

The testbed used in our experiments is made of 308 tweets written in *Arabic* and *English* language, which have been manually selected from an on-line repository[2] and labeled either as "Normal" or "Terrorist". It is worth noting that all tweets in the testbed are related to ISIS, hence the task of the framework is differentiating between those tweets which are supporting Jihad, and those reporting news related to ISIS, war in Iraq and Syria. Since all tweets are related to the same topic, they all share the same set of words, even if with different frequencies. Afterward, a set of 19 words in Arabic and English containing the common words that ISIS supporters generally utilize is provided and depicted in Figure 6, where the font size is proportional to the word frequency in the analyzed tweets. These words compose hence the x_w set provided by the data provider, whilst the analyzed tweets constitute the x_d, one at a time. As explained in Section III, the words in x_w and the words in x_d are turned to hash values using the SHA-1 hash function. The extracted vectors are hence passed to a classifier in the data analyzer, which will differentiate the vectors between the ones belonging to terrorist tweets and normal ones. The dataset is composed of 160 normal tweets and 148 terrorist tweet, hence the dataset is almost balanced, divided in Arabic and English tweets. Table I reports the comparison between 7 well known classifiers, in terms of correctly and wrongly classified instances for both classes, reporting thus True Positive Rate (TPR, percentage of Terrorist tweet correctly classified), False Positive Rate (FPR, percentage of Normal tweets wrongly classified), True Negative Rate (TNR, percentage of Normal tweets correctly

[2]www.kaggle.com

TABLE I: Classification results evaluated on 5-fold cross validation on ISIS dataset.

Algorithm	**K-star**		**C4.5**		**NaiveBayes**		**SVM**		**KNN**		**MLP**		**RandomForest**	
Classification	√	×	√	×	√	×	√	×	√	×	√	×	√	×
Terrorist	0.818	0	0.858	0	0.926	0	0.878	0	1	0	0.973	0	1	0
Normal	1	0.182	1	0.142	1	0.074	1	0.122	1	0	1	0.027	1	0
Accuracy	0.912		0.931		0.964		0.941		1.000		0.987		1.000	

Fig. 6: A Set of ISIS Supporters Words

classified), False Negative Rate (FNR, percentage of Terrorist tweets wrongly classified) and overall accuracy, calculated through 5-fold evaluations. As shown, all classifiers show a very high TNR, as a consequence of the small majority of Normal tweets on the terrorist ones and of the reduced frequency of critical words in Normal tweets, which makes them easy to identify. Among the classifiers, the ones showing a higher accuracy are the NaiveBayes the KNN and the RandomForest. Both NaiveBayes and KNN are classifiers whose training process has a low computational complexity, and exploit distribution in feature space as discriminant, showing thus that vector of same classes are effectively similar.

B. Privacy Preserving Spam Bot Detection

Spam emails constitute a notorious and consistent problem still far from being solved. In 2013, for example, out of the daily 191.4 billions of emails sent worldwide in average [22], more than 70% are spam emails that cause several problems, spanning from direct financial losses, to misuses of Internet traffic, storage space and computational power. The problem of spam email becomes more severe when they are served as a tool to perpetrate different cybercrimes, such as phishing, malware distribution, or social engineering-based frauds. Generally, these kinds of attacks are launched with the aim of making the users believe that they are communicating with a trusted entity for the purpose of stealing account information, login credentials, and identity information in general. The final goal is generally to steal money, either directly by withdrawing from their account or indirectly by forcing them to pay themselves (e.g. ransomware) [26].
Nonetheless, generally spammers have the minimum risk to be prosecuted or be caught, since they generally hide themselves behind complex networks of infected devices that send spam emails against their user will or awareness. Such networks, named botnets, are able to generate, at the spammer command, millions of email directed toward other recipients (victims), unawarely protecting the spammer from being identified and prosecuted. However, identification of devices which are part of a botnet and consequent removal of malicious code from the device, should help in strongly limiting the amount of generated spam traffic. Considering that spammers often send spam emails in campaigns [27], i.e. emails showing the same or similar format in structure and text, a targeted analysis looking for specific words and their frequency may give a greater insight about botnet structure [29]. Hence, the bag-of-words methodology can be a powerful tool to automatically recognize if an email is part of a specific spam campaign. However, finding the devices that are part of a botnet is a challenging task. In fact, source email address can be changed programmatically by the spammer, also IPs are dynamically changed periodically and are thus difficult to track. However, information on the email sender can be found either in the device outbox, or at telephony provider's side, which by law must keep record of outgoing traffic for each subscriber. Still, such an analysis brings several privacy implications, since the data analyzer should have access to all the outgoing user emails. To mitigate this issue, it is possible to apply our proposed framework, performing analysis on users emails (x_d), using a set of words which is representative of a specific spam campaign (x_w), exploiting HE to not disclose email contents and harm users' privacy. For the sake of simplicity, we will focus on a single specific spam campaign, where the spammer advertises a *Lottery* campaign.

Fig. 7: Words extracted to detect lottery campaigns.

TABLE II: Classification results evaluated on 5-fold cross validation on Lottery spam dataset.

Algorithm	**K-star**		**C4.5**		**NaiveBayes**		**SVM**		**KNN**		**MLP**		**RandomForest**	
Classification	√	×	√	×	√	×	√	×	√	×	√	×	√	×
LOTTERY	0.969	0	0.969	0	1	0.012	1	0	0.969	0	1	0	0.969	0
GOOD	1	0.031	1	0.031	0.988	0	1	0	1	0.031	1	0	1	0.031
Total	0.991	0.008	0.991	0.008	0.991	0.008	0.991	0.008	0.898	0.273	1	0	0.991	0.008

The 17 relevant words which build the x_w vector are depicted in Figure 7, again with the font size representing the frequency of the specific word. The testbed, used also to select the words in x_w is made of 236 real emails, with 64 emails belonging to a lottery spam campaign and 172 genuine emails extracted from real user outboxes. We have evaluated the capability of 7 classifiers, the same used for the previous use case, to discern between *Good* emails and the ones belonging to the *Lottery* spam campaign. Results and comparison in accuracy between the classifiers are reported in Table II, following the same schema of the previous example, i.e. True Positive Rate (TPR, the percentage of Lottery emails correctly classified), False Positive Rate (FPR, the percentage of Good emails wrongly classified), True Negative Rate (TNR, the percentage of Good emails correctly classified), False Negative Rate (FNR, the percentage of Lottery emails wrongly classified) and overall accuracy.
As shown, also in this case, all classifiers show a very high TNR. This is a consequence of the typical behavior of several classifiers which, to minimize the overall error, tend to maximize the probability of classifying an element as belonging to the majority class. It must be noticed, in fact, that the two classes are not balanced, still it is possible to infer by the overall accuracy that classifiers do not suffer from overfitting with the provided dataset. the small number of good emails classified as Lottery and because of the reduced frequency of critical words in Good emails, which makes them easy to identify. NaiveBayes is the classifier that shows the higher accuracy, which it has already been proven in the literature to be an effective tool for spam filtering [18]. It is worth noting that these classifiers in our experiments misclassified very few Good emails as Lottery, hence it becomes unlikely that a good user will have privacy violation due to the result of the analysis performed by the proposed framework.

C. Dataset Dimensioning

The process of manual analysis and labeling of both tweets and emails is largely time consuming, which imposes a limitation on the dataset size. However, the dataset has been correctly dimensioned, ensuring the lack of duplicates and with enough representative of the two classes. Thus, standard dimensioning techniques have been used, for testbed datasets.

TABLE III: Datasets specifications.

	Elements	W/E	Good	Bad	Features
Tweets	308 (77)	32	160	148	19
E-mails	236 (59)	147	172	64	17

The original datasets are made of respectively 77 tweets and 59 real emails. To increase the size of dataset in the classification phase, the SMOTE oversampling algorithm has been used to increase the size of both datasets, generating additional non duplicate vectors. The details on the dataset dimensioning are reported in Table III, where the first column represents the amount of elements after oversampling (in brackets the original size), the second columns represents the number of words per element (email or tweet), followed by the number of good and malicious elements, and the number of features. A general rule to assess the minimum size for a training set is to dimension it as six times the number of used features [30]. In our analysis the number of features is given by the number of words constituting the dictionary of words, hence for tweet dataset and spam email dataset, we have respectively the dictionaries of sizes 19 and 17, and testbed datasets of sizes 308 and 236 words. This means that the datasets size satisfies the condition of being reliable, i.e. $6 \times 19 < 308$ and $6 \times 17 < 236$.

D. Performance Analysis

Full homomorphic encryption is a technique which ensures high level of confidentiality, given a correct selection of parameters, still the complexity required by the encryption and computation on encrypted data, imposes a massive overhead. In the current work, full homomorphic encryption is applied to perform word search in two dictionaries, respectively of 17 and 19 words. We have measured the performance of the proposed system on the two datasets, to extract the average analysis time per element. In particular, it has been measured the amount of time to generate a single vector of features, from the beginning of encryption on the side of data provider, to the result decryption on the data analyzer. Experiments have been run on a pc with Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz, four cores and 16 GB of RAM.

The performance results are presented in Table IV and Table V, where the first column reports the security level yield by the BGV algorithm for the chosen value of the m parameter. The security level *sec*, which quantifies the needed effort to break the cipher as 2^{sec} operations, is dependent from the value of m, increasing almost linearly for high levels of this variable. The values used in our experiments for m are 6023, 7023 and 8023, where the first value has been chosen by automatic configuration of the optimal parameters through the `findM()` routine and a value of 1000 has been added twice to increase the security level.

Reported times are expressed in seconds and specify the amount of time necessary to find a single word in the dic-

TABLE IV: Performance results for tweet dataset (time in s)

S.L.	Pre-Ordering			Quadratic Search		
	Word	Elem	Total	Word	Elem	Total
116	31	1147	90379	225	7215	560978
37	38	1406	110305	287	9213	716673
12	29	1073	82687	153	4884	378925

tionary, to analyze a complete element, i.e. all the words of an email or tweet in order to generate a vector, and the time needed to complete the operation on the total dataset. Both experiments have been performed with two different algorithms for word search in the dictionary: Pre-Ordering and Quadratic Search. Quadratic search indicates the algorithm for which every word is compared with all possible words in the dictionary, which yields a complexity of $O(n^2)$. The

TABLE V: Performance results for the email dataset

S.L.	Pre-Ordering			Quadratic Search		
	Word	Elem	Total	Word	Elem	Total
116	32	4704	282887	201	29609	1755852
37	40	5880	345255	258	38019	2243174
12	30	4410	258811	136	20102	1186030

Pre-Ordering approach strongly reduces the complexity, by ordering the results of the hash functions before encryption, both on data provider and data analyzer side. This allows to stop the search every time the sign of the homomorphic subtraction result changes from negative to positive, helping to search the rest of the dictionary. This results in a noticeable increase of performance and in a reduction of complexity to $O(n \cdot log(n))$. The time required by a single word comparison is of 0.5 seconds, including the operation of encryption and decryption. All tests have been run four times in parallel on separate threads, and the results come as an average of the four experiments.

As shown in Tables IV and V, the overall analysis time does not directly depends from the security level and the best compromise is obtained with the highest security level. However, as expected, the analysis time is extremely high, i.e. 19 minutes to analyze a single tweet and 78 minutes to analyze an email, and not feasible for large datasets, having only an average computational power. However, it is noticeable that the proposed architecture can be completely parallelized, increasing the number of homomorphic engines

E. Security Analysis

The proposed framework, exploiting homomorphic encryption ensures that all exchanged messages among the three components are constantly encrypted through the full homomorphic encryption algorithm BGV [2], where the algorithm parameters have been set to have a *security level* of 116. This ensures that an attacker attempting to perform a known cypher-text attack, has to perform 2^{116} operations to break the cipher, which is computationally infeasible for most attackers. In particular, the data provider is the only one to have access to plaintext of the various x_d. The homomorphic engine receives the x_d vector encrypted with the public key of the data analyzer, hence, it is not able to decrypt the message. This makes the cypher-text attack impracticable, both by the homomorphic engine and by an external attacker eventually eavesdropping the message. The same applies to the x_w vector which is only visible to the data analyzer. The dictionary attack would be not effective as well, in fact, the BGV algorithm is not prone to this kind of attack, thanks to the addition of random padding bytes, which generates different cypher-texts for the same plain-text, making impossible to create a dictionary cypher-text to plain-text.

It is worth noting that classifiers did not misclassify in our experiments normal tweets as malicious, hence it is extremely unlikely that a normal user will have privacy violation due to the result of the analysis performed by the proposed framework. Moreover, it is possible to protect the system from record leakage attack by exploiting *differential privacy* measures, like the one proposed in [5], directly on the data provider, to minimize the risk of a successful record-leakage attack.

V. Related Work

Hummingbird is one of the very first attempts in constructing a privacy enhanced micro-blogging Online Social Network architecture [4]. In this study, Tweeters control who can access to their tweets while enforcing fine-grained access control. That is, a tweeter authorizes followers to read only tweets with specific hashtags. However, the proposed approach works through the control of users on their own information sharing and it is not useful in the case that an external organization requires to securely analyze the contents. A privacy preserving approach is proposed in [16], for multi document summarization. The proposed methodology enables other parties to get the summaries of each other documents without learning some thing else about the original documents' content. A hashing technique, named *Secure Binary Embeddings*, is used to convert the documents into bit strings, which is used to approximate the distances of text documents. Still, when data are transformed to hash values, it is possible to detect the original text, and the privacy gain is not as high as encryption techniques. In [23] the Levenshtein distance of two strings which two parties own is computed securely with the use of a proposed asymmetric two-party protocol. However, the proposed technique is set between two parties, and can not be practical in the problem of the pile of texts collected from large number of sources.

A privacy preserving multi keyword text search scheme, with the use of similarity-based ranking methodology, has been proposed in [28]. For supporting multi keyword search and search result ranking on the base of word frequency and vector space model, an index calculated through cosine similarity metric, is attributed to get the higher search result accuracy. For addressing privacy leakage, two secure index schemes, named Ciphertext and background model, are applied.

Aggarwal and Yu [1] formalized an anonymity model for the sketch-based approach, and utilized it to construct sketch-based privacy-preserving representations of the original text. The sketch-based approach reduces the dimensionality of the data by generating a new representation with a much smaller number of features, where each one uses a different set of random weights to produce a weighted sum of the original feature values. This technique is effective when the dataset is high dimensional and the data is sparse [1]. Saygin et al [24] sanitize text documents by removing sensitive information or potential linking that can associate an individual person to the sensitive information in a document. However, the privacy preserving techniques, which are based on the reduction approaches, are not as secure as encrypted data, since published sanitized data may still violate the privacy.

VI. Conclusions and Future Work

Privacy-preserving text mining is a challenging task, since performing analysis on data, without disclosing the privacy sensitive information, is not always viable. This work has presented a preliminary framework and the related protocol, based on the interactions of three agents, to perform word frequency analysis on privacy sensitive texts, exploiting homomorphic encryption and bag-of-words classification. The presented framework allows to perform data analysis without disclosing privacy sensitive information neither to the data analyzer, nor to the homomorphic engine itself. We have reported two real use cases, in which the word analysis is relevant to detect possible critical situations (i.e. terrorist tweets and bot infected devices), still without disclosing any sensitive information on the analyzed text or its owner. As future work, we are planning to improve the performance of the proposed framework by using GPU-based computation and parallel architecture. Moreover, a logic to verify the effective non disclosure of private information after the analysis, or a measure to quantify it (eventually based on differential privacy), will be introduced to make aware the data provider of eventual privacy leaks.

References

[1] C. C. Aggarwal and P. S. Yu. *On Privacy-Preservation of Text and Sparse Binary Data with Sketches*, pages 57–67. 2007.

[2] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA*, pages 309–325, 2012.

[3] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, pages 509–516, Menlo Park, CA, USA, 1998.

[4] E. D. Cristofaro, C. Soriente, G. Tsudik, and A. Williams. Hummingbird: Privacy at the time of twitter. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 285–299. IEEE Computer Society, 2012.

[5] C. Dwork. *Differential Privacy*, pages 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[6] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012.

[7] R. Feldman and J. Sanger. *Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, New York, NY, USA, 2006.

[8] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

[9] P. Giudici and S. Figini. *Market Basket Analysis*, pages 175–191. John Wiley and Sons, Ltd, 2009.

[10] S. Halevi and V. Shoup. Algorithms in helib. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 554–571, 2014.

[11] Z. S. Harris. Distributional structure. pages 3–22, 1981.

[12] T. Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 137–142. Springer-Verlag, 1998.

[13] W. Magdy, K. Darwish, and I. Weber. Failedrevolutions: Using twitter to study the antecedents of isis support. *CoRR*, abs/1503.02401, 2015.

[14] M. B. Malik, M. A. Ghazi, and R. Ali. Privacy preserving data mining techniques: Current scenario and future prospects. In *2012 Third International Conference on Computer and Communication Technology*, pages 26–32, Nov 2012.

[15] F. Martinelli, A. Saracino, and M. Sheikhalishahi. Modeling privacy aware information sharing systems: A formal and general approach. In *15th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, 2016.

[16] L. Marujo, J. Portelo, W. Ling, D. M. de Matos, J. P. Neto, A. Gershman, J. G. Carbonell, I. Trancoso, and B. Raj. Privacy-preserving multi-document summarization. *CoRR*, 2015.

[17] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification, 1998.

[18] V. Metsis and V. Metsis. Spam filtering with naive bayes – which naive bayes? In *Third Conference on Email and Anti-Spam (CEAS*, 2006.

[19] K. Nigam, A. K. Mccallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2):103–134, 2000.

[20] S. R. M. Oliveira and O. R. Zaïane. Privacy preserving frequent itemset mining. In *Proceedings of the IEEE International Conference on Privacy, Security and Data Mining - Volume 14*, CRPIT '14, pages 43–54, Darlinghurst, Australia, Australia, 2002. Australian Computer Society, Inc.

[21] M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.

[22] S. Radicati. Email statistics report 2013-2017, 2013.

[23] S. Rane and W. Sun. Privacy preserving string comparisons based on levenshtein distance. In *2010 IEEE International Workshop on Information Forensics and Security*, 2010.

[24] H.-T. D. SAYGIN, Y. and G. TUR. *Web and Information Security*. 2006.

[25] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34:1–47, 2002.

[26] M. Sheikhalishahi, A. Saracino, M. Mejri, N. Tawbi, and F. Martinelli. Digital waste sorting: A goal-based, self-learning approach to label spam email campaigns. In *Security and Trust Management - 11th International Workshop, STM 2015, Vienna, Austria, September 21-22, 2015, Proceedings*, pages 3–19, 2015.

[27] M. Sheikhalishahi, A. Saracino, M. Mejri, N. Tawbi, and F. Martinelli. Fast and effective clustering of spam emails based on structural similarity. In *8th International Symposium on Foundations and Practice of Security*, 2015.

[28] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li. Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ASIA CCS '13, pages 71–82, New York, NY, USA, 2013. ACM.

[29] K. Tretyakov. Machine learning techniques in spam filtering. In *Data Mining Problem-oriented Seminar, MTAT*, volume 3, pages 60–79. Citeseer, 2004.

[30] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001.