Preventing Hidden Information Leaks Using Author Attribution Methods and Neural Networks

Alexander Khazagarov, Alisa Vorobeva, Viktoriia Korzhuk
ITMO University
St. Petersburg, Russia
akhazagarov@gmail.com, vorobeva@itmo.ru, vmkorzhuk@itmo.ru

***Abstract*—This paper addresses the problem of hidden information leakage detection through the use of text steganography. Presented comparative research results show how to perform this task by detecting changes in user's writing styles using neural networks and various types of text features. The framework for hidden leakages detection based on discovering changes in the author's writing style with deep neural networks (RNN, LSTM, GRU, CNN) is proposed. A series of experiments on text corpus containing Russian online texts were carried out to evaluate hidden leakages detection accuracy. The experiments showed that the LSTM and character 4-grams together allow achieving the accuracy of 87%. Text preprocessing significantly decreases accuracy which is also shown.**

I. Introduction

Any business of modern enterprise comes hand in hand with processing, transferring, and storing various types of digital data. More and more companies are using clouds, electronic document management, and Enterprise Resource Planning (ERP) systems. Modern communication allows exchanging hundreds of gigabytes of data within seconds. Usage of instant messaging (IM), e-mails, and social networks make data exchange even more accessible. Not only frequency of leakages of sensible information increase, but its nature changes, a completely new leakage type is arising [1]. Steganography allows hiding the fact of information transfer. At the same time, modern text steganography methods make changes imperceptible to the human eye and information security specialists may not reveal a hidden leak. Detection of text steganographic tools is one of the most challenging tasks. This task is complicated by the availability of various steganographic methods, and the usage of each cannot be detected.

Using steganography techniques allows confidential information to be sent outside the secure perimeter under the guise of a regular message file. Also, the application of text steganography to an ordinary document and the takeout of a printed document outside the protected boundary is not difficult in many enterprises.

The long-term goal of modern steganalysis is to develop universal methods, that can detect messages hidden with almost all steganographic tools. Most of the previous research is focused on detecting one type of steganography. Existing universal steganalysis methods are more complex and less efficient than specialized, aimed at detecting specific types of message hiding. It should be noted that although the problem of hidden leakage is relevant, only a small number of universal steganalysis methods exist. Despite that, progressive universal methods are more practical oriented and have more practical value.

The most effective universal steganalysis method is SARC (Steganography Analysis and Research Center) [2]. According to 2019 statistics, its average accuracy is 80%, which is rather low. Thus, relatively low accuracy of detection of hidden leakages can be identified as a major problem.

Neural networks (NN) are a very promising alternative to classical steganalysis methods [3-5] Text steganalysis is not an exception. However deep learning universal steganalysis methods use a training set including two types of samples: normal and stego texts. Models are trained to detect texts with messages hidden with predefined steganographic methods. It is alike the signature-based approach in computer virology. The main limitation is that it will not be able to detect new steganographic methods, that were not presented in training data.

The proposed approach is an alternative to the methods mentioned above, it is some kind of anomaly detection. It is supposed that the use of text steganography tools changes the author's writing style. The use of linguistic identification or author attribution methods to detect text steganography has not been studied yet, however, it can be useful for hidden information leakage detection and disclosing cybercrimes associated with it.

A. Problem description

The problem of detecting the usage of text steganography and hidden information leakage can be described as follow. Given a set of users $U = u_1, ..., u_k$ and their text messages $T = t_1, ..., t_m$, where m - number of messages and k is number of users.

One of the users - $u_a \in U$ is creating the hidden communication channel for transmitting the secret message (t_{secret}) to external user $u_b \notin U$. We should find algorithm – a, that is able to detect text, containing t_{secret}, among all texts of u_a, or in other words to discover the fact of hidden message transmission - t_{secret} in the cover text message t_{cover}.

B. Previous research

As it was mentioned before, all studies of steganalysis can be divided into two areas: universal algorithms and specialized algorithms. The current trend in steganalysis is to develop universal methods. Universal steganalysis implies finding common properties and external features while studying a wide range of existing steganography algorithms [6-10]. The most popular approach here is supervised learning. Model is trained on the dataset including both steganographic text samples and clear (normal) texts. Based on this data, the system finds the best classification rule [6,25].

In article [11] Authors use Recurrent neural network (RNN) to extract statistical patterns feature distribution differences and then classify those features into cover text and stego text categories. This model can make use of the subtle distribution difference of the features to estimate the capacity of the hidden information inside.

In article [12], a linguistic steganography detection algorithm using a statistical language model is presented. The result of the experiment shows when the text segment size is 2 kB (kilobyte) and 5 kB, the detecting accuracies are found to be 93.9% and 96.3% respectively.

There are also other types of training methods like parametric statistical modeling, but they are less popular.

In this work, we study the possibility to detect hidden leakages while detecting changes in user's writing style using neural networks and various types of features. Also, we propose a framework for hidden leakages detection and study accuracy on a dataset including texts with the application of various steganographic methods: Unicode stego tags, advanced stego tags, Butt Ugly Latin Wide, trailing spaces etc. Different architectures of deep neural networks are analyzed as well as different feature extraction methods.

II. Preventing hidden information leaks with author linguistic identification methods

A. Approach to prevent hidden information leaks with author linguistic identification methods

Every human has his distinguishable writing style that is subconsciously included in the author's text messages. If we can extract some writing-style features of the messages that are inherent to a particular user, we are then able to discover the changes in the author's style. If steganography is applied to the text, the original text is changed and this leads to the author's writing-style alteration [26]. It is supposed that author attribution or linguistic user authentication methods could be applied to discover the hidden leakages.

With the proposed approach, the detection of text steganography is narrowed to the detection of author style alteration. Thus, the task transforms into binary classification.

Given a set of texts $T = t_1, ..., t_m$ of authors U, where m - number of texts. The author u_i can be presented as a subset $T_i \subset T$.

To train a classifier a_i that predicts if the writing style of the new text message t_{new} is the same as in previous texts of u_i, a T_i could be used. The trained algorithm outputs 1 if there is no author style alteration for u_i and 0 if the author's writing style has changed.

The problem of author identification is old enough, but it is remaining relevant. The availability of machine learning methods allowed us to look at the linguistic identification problem from a new perspective. In recent years, the study in the field of linguistic identification has significantly advanced [13]. The main reasons are the development and popularization of various machine learning techniques. However, most of the existing works are theoretically and practically focused only on texts in English or Germanic languages.

In most studies classical machine learning methods (decision trees, k-means, support vector machine, random forest, etc.) were used to attribute the author of the short messages [13]. In some studies, authors have found that the use of neural networks in the task of author identification from a small set of candidate-authors is inefficient because of high memory consumption and approximately the same accuracy as the methods mentioned above.

In the article [14] is discussed author attribution using deep neural networks. Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), Long Short-Term Memory Network (LSTM), and several hybrid models were compared. In experiments were used datasets containing texts in Russian from several websites. CNN and n-grams showed the best result.

In various articles, there is a tendency to use characters n-gram [13]. Characters n-gram gives a representation of author stylistics, grammar, punctuation, and text topic. Most modern researchers give their preference to character n-grams. This approach receives information about the peculiarities of the author's style through the most popular words, grammar, punctuation [13-19]. In articles [20,21,24] an ability to identify the user based on his email texts is discussed. The accuracy of classification significantly increases when special characteristics are combined with classical ones.

B. Text vectorization for neural networks

A neural network works only with numerical features (this comes from its mathematical model). To be classified by NN texts should be represented as numerical data. This process is known as vectorization – converting string features into numerical features. In the first step, tokenization is performed: the whole text is splitting up into chunks (words, symbols, or sentences). The next step is to perform vectorization based on the tokenization tree. There are several different ways for vectorization:

- Numerical coding – each token is matched to some unique identity.
- One hot encoding – representation in the form of the sparse vector that consists only of zeroes and single number one. The length of the vector is equal to the number of tokens. All vector values are equal to zero except that one that is corresponding to the desired token.
- Embedding (dense vector) – each token is set with a vector in which any number can be used so that it is denser than 'One hot encoding'.
- N-gram is a set of tokens from 1 to N. N-gram divides the text into continuous groups of n tokens making a denser vector representation. N-gram is one of the methods for

extraction of feature set from the text and is usually used with classical methods to analyze text.

Neural networks usually do not need some special feature set extraction. If we have data set that is large enough then the neural network can find all the required patterns on its own. But in [22] was proposed feature space that showed a quite high author attribution accuracy for texts in Russian. In experiments, we will study if it is possible to use it as an input vector for NN to detect hidden leakages. The feature set includes 445 features listed in Table I.

TABLE I. FEATURES OF THE ELECTRONIC MESSAGES

Features group	Features
Lexical symbol level	1. message length in symbols. 2. frequency of the capital letters. 3. frequency of the letter symbols. 4. frequency of digits. 5. frequency of spaces. 6. tabulation frequency. 7. frequency of the letters with umlaut.
Lexical word level	8. total word count. 9. mean length of words in symbols. 10. total sentence count. 11. mean sentence length in symbols. 12. reserved words frequency. 13. abbrev. usage frequency. 14-432. frequency of functional words.
Syntax level	433. frequency of punctuation. 434-444. frequency for each punctuation symbol. 445. total frequency of functional words.

III. METHOD AND FRAMEWORK FOR PREVENTING HIDDEN INFORMATION LEAKS

The method for countering hidden information leaks includes three stages. The flow chart is shown in Fig. 1.

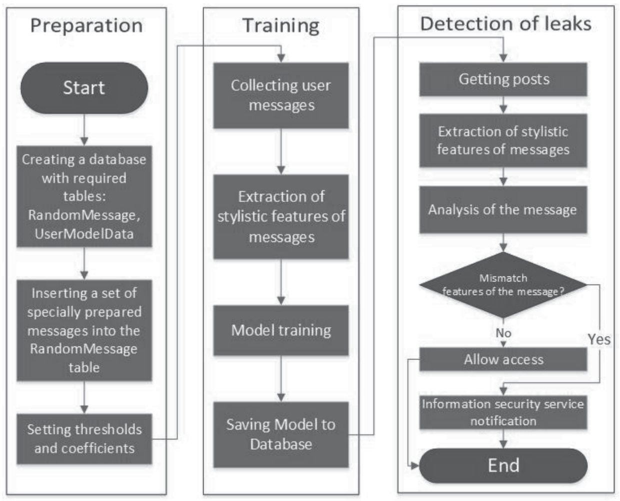
Preparation
Start
Creating a database with required tables: RandomMessage, UserModelData
Inserting a set of specially prepared messages into the RandomMessage table
Setting thresholds and coefficients
Training
Collecting user messages
Extraction of stylistic features of messages
Model training
Saving Model to Database
Detection of leaks
Getting posts
Extraction of stylistic features of messages
Analysis of the message
Mismatch features of the message?
No
Yes
Allow access
Information security service notification
End

Fig. 1. Flow chart of preventing hidden information leaks method

The preparatory stage, at which the database with the necessary tables is creating. The stage of training the user model based on his messages. The stage of detecting hidden leakages and using steganographic methods. It is assumed that it is possible to uniquely identify users.

Fig. 2 shows the general scheme of hidden leakages detection framework implementation. Arrow 1 shows the message transferring for the leakages detection, and arrow 2 shows the result of the analysis. On the user's workstations should be installed messages interception software, that captures instant messaging (IM) and then sends texts to the server for further analysis. Alert messages are sent to the cybersecurity specialists if the writing style of the user significantly changes, otherwise, if the system had not detected information message is passed to the recipient and stored in the database.

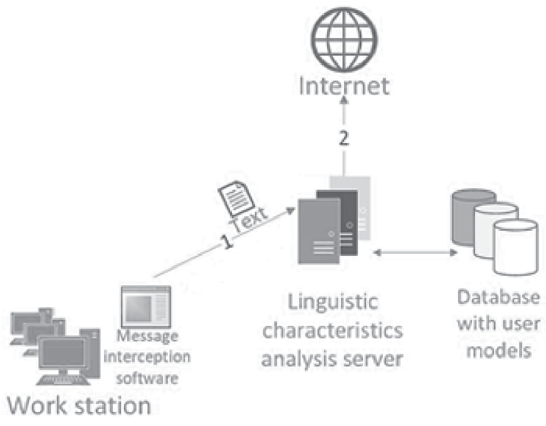
Internet
2
Text
1
Message interception software
Work station
Linguistic characteristics analysis server
Database with user models

Fig. 2. General scheme of hidden leakages detection framework implementation

A. Preparatory stage

At the preparatory stage, the RandomMessage and UserModelData tables are created, the structure of which is shown in Tables II and III, respectively. Various initial values are set at this stage: length message, trigger accuracy for possible leakage and sensitivity coefficient.

TABLE II. RANDOMMESSAGE TABLE STRUCTURE

Name	Type	Size	Description
Id	int	-	ID, primary key
Message	String	Max	Message

TABLE III. USERMODELDATA TABLE STRUCTURE

Name	Type	Size	Description
Id	int	-	ID, primary key
UserLogin	Nvachar	255	User login, foreign key
StateNetwork	-	-	Neural network model
Accuracy	Float	-	Accuracy

Messages should be fixed length (l). We recommend to set value of l based on the analysis of messages length (in symbols) frequencies. In this study we use l=600 symbols as such length covers 85% of all messages. If the message is longer it is split into several with a maximum length of 600 symbols. For example, if the message length is 957 symbols it is divided into two: the first one containing 600, and the second containing 357 symbols (the remaining 243 symbols are filled with zeroes). Appropriately selected l value allows to use less number of insignificant symbols, in our case it's zero.

Messages from various users, stripped of markup, in Russian and English languages are added in the RandomMessage table. All messages should be cleared from markup, images, and other attachments. The minimum length of the message should be at least 300 characters.

B. Training stage

For each user, the system must train a model based on his writing style characteristics. For training, the system should use only sent text messages. Training model of hidden leakages detection algorithm is presented below.

Algorithm 1 Training model of hidden leakages detection

```
1: Create a list of type MessUX.
2: Select all messages sent by the user.
3: Clear all messages from HTML tags, pictures, and tables.
   If the message is a “reply to”, then only the reply is saved.
4: For each message check for the correct length. If the
   message is more than 600 characters, then it is split into
   several messages, each of which is a multiple of 600
   characters. If the message is less than 600 characters, then
   it is padded with leading zeros so that the last character of
   the message is at position 599.
5: Vectorize message, depending on the vectorization and the
   feature extraction methods.
6: For each message, an instance of an object of type MessUX
   is created, to which a unique key is generated in the id value,
   the message takes the message value, vector the vector
   value, and MuFlag is true.
7: Each message is added to an instance of the object.
8: Random messages from the table “RandomMessage” that
   do not belong to the user are added to the list in a ratio of
   5: 100, where 5 are user messages. (MuFlag – false, vector
   – vector message).
9: Shuffle list.
10: Train model based on the values of the Vector and
    MuFlag fields.
11: If the accuracy of the model is less than 70%, then
    the algorithm ends. The training will be restarted after an
    increase in the number of relevant user messages by 10
12: The UserModelData entity is added to the database table
    with values (userLogin – user login, StateNetwork – saved
    user model, Accuracy – model accuracy).
```

For each user, the system must train a model based on his writing style characteristics. For training, the system should use only sent text messages.

The system starts training only if there are at least 50 messages per user. Parameters such as model tolerance accuracy can be set locally. The maximum message length can also be set depending on the specifics of the work. For training, the MessUX class is used, the structure of which is shown in Table IV.

TABLE IV. USERMODEL TABLE STRUCTURE

Name	Type	Size	Description
Id	int	-	ID, primary key
Message	String	600	User's message
Vector	-	-	Vector message representation
MuFlag	Bool	-	Indicate if the message belongs to the user

C. Detection of leaks

At the second stage, it is detected whether text steganography was applied to the text of the sent message of the user. The message is checked for the correct length. If the message is more than 600 characters, then it is split into several messages, each of which consists of less than 600 characters. If the message is less than 600 characters, then it is padded with leading zeros so that the last character of the message has position 599. In algorithm 2 the leaked detection process is described.

Algorithm 2 Detection of leaks

```
1: Based on the user, the UserModelData object is loaded and
   the model is initialized based on the StateNetwork property.
2: A list of type MessUX is created.
3: The message is presented in the form of a vector, depending
   on the vectorization method and the method of feature
   extraction
4: For each message, an instance of an object of type MessUX
   is created, to which a unique key is generated in the id
   value, the message takes the message value, vector the
   vector value, and MuFlag is true. Each object is added to
   the MessUX list.
5: Each message goes to the trained neural network, after
   which the result of the calculation is written to the object.
   The result of the calculation is a boolean variable.
6: Next, the result is calculated by the formula $(\sum n * k)/(count(n))$,
   where n is the results of message calculations.
7: If the probability of a mismatch is less than the threshold
   value ($P$), then the information security service is notified.
```

In the algorithm above is used following notation: P is a threshold value that determines whether a message does not contain hidden leakage, the default value is 0.7, k is the multiplying factor for the algorithm to trigger, n is the result of the model for each message (in 600 characters).

D. Extended model training

If it turns out that a false alarm has occurred, as a result of the investigation of an incident by the security service, then there is an opportunity to train the model. To do this, the

UserModelData object is unloaded for a specific user and the system is retrained using the specified message.

Eight possible resulting Key Quality Indicators (KQI) are identified: identification accuracy, False Rejection Rate (FRR), False Acceptance Rate (FAR), identification speed, error, training time, resource intensity, and quality indicator. For the study, both standard and field-specific indicators were selected.

E. Neural network configuration

In this work, we use an RNN is a class of neural networks that are good for modeling data sequences such as time series or natural language [22-23]. We will consider RNN, LSTM, GRU and CNN networks.

RNN is a classic recurrent neural network that contains feedback and allows you to store information. This network has a significant disadvantage such as a vanishing gradient. As the length of the analyzed text increases, the network loses its ability to connect information. Therefore, information cannot be stored for a long time [23].

Long short-term memory (LSTM) is a kind of recurrent neural network architecture capable of learning long-term dependencies.

Gated Recurrent Units (GRU) is one of the types of recurrent neural networks, similar in structure to LSTM, but it has fewer parameters, there is no input gate.

The structure of the neural network is described below:

- Embedding
- RNN/LSTM/GRU – 200
- RNN/LSTM/GRU – 400
- BatchNormalization
- RNN/LSTM/GRU – 100
- BatchNormalization
- Dense

IV. Experiments and results

A. Text corpus and dataset

To examine the accuracy of the proposed framework and consumed resources were carried out series of experiments. In experiments, we used text corpus including texts of various topics both in Russian and English. For each text author is set. The text corpus consists of open datasets and collected data:

- Habr – a set of articles by various authors on the topic of information technology.
- VK – a set of posts of users from a social network.
- Echo of Moscow – a collection of blogs by various journalists, politicians, economists, writers, and historians on various topics.
- LiveJournal is a collection of blogs from different users on different topics. Often used for online diaries.

In total, the corpus contains 189,328 texts for 1,200 authors. Fig. 3 shows the distribution of the text's length.

It is recommended to use ten or more relevant texts per author. Accordingly, the more texts the author has, the more accurately it is possible to identify his characteristics. Fig. 4 shows the distribution of the number of texts and authors.

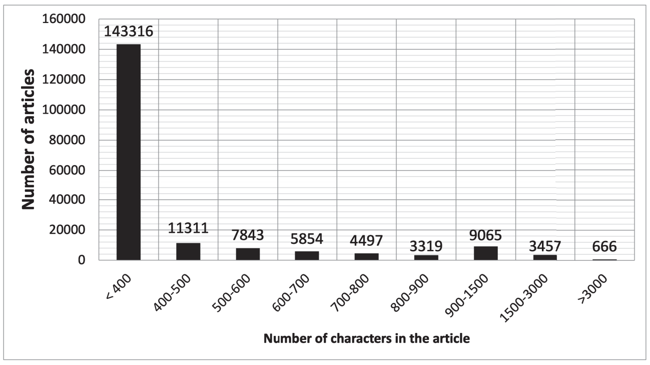

Fig. 3. Distribution of texts length

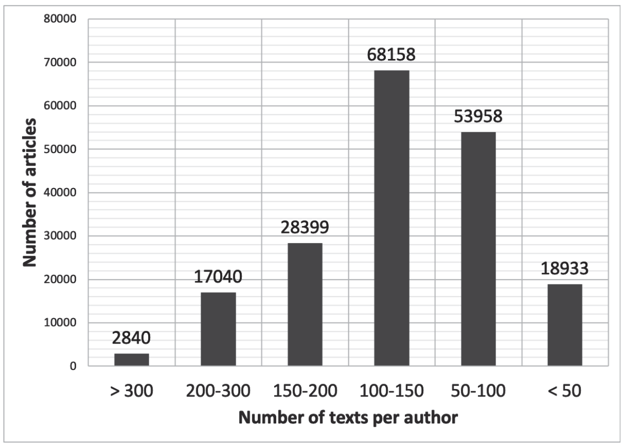

Fig. 4. Distribution of the number of texts per author

Following Fig. 3 and 4, all texts in the corpus can be used for the task of countering hidden leakages of information.

The final dataset includes 50 000 texts, with a minimum length of 300 characters. For training NN we have used texts in proportion 1:50 (text of the author u_a: texts of other authors). For several texts of each author steganographic methods were applied: Unicode stego tags, advanced stego tags, Butt Ugly Latin Wide, trailing spaces, and syntax methods.

B. Study influence of text preprocessing on the accuracy of leakages detection

Each person has a different writing style. This includes usually the most common word series, punctuation marks, and even spelling errors. Table V shows the results of the experiment showing the effect of using text preprocessing on the accuracy of determining information leakage. 4-grams are used as a feature extraction method.

TABLE V. Impact of text pre-processing on leak detection accuracy

Type of text	Accuracy,%			
	RNN	LSTM	GRU	CNN
With pre-processing	34	43	40	40
Without pre-processing	73	87	85	80

Many text recognition tasks using neural networks require eliminating insignificant words and applying normalization for each word. However, shown in Table V, text preprocessing decreases leak detection accuracy. In this case, the system works almost at random. Preprocessing of the text removes most of the markers that are left by the steganography algorithms. Especially text preprocessing is detrimental to methods based on formatting, adding punctuation characters, and trailing spaces. Let's consider the effect of text preprocessing for binary user classification. The experimental results are shown in Table VI.

TABLE VI. INFLUENCE OF TEXT PREPROCESSING ON THE ACCURACY OF BINARY CLASSIFICATION

Type of text	Accuracy,%		
	RNN	LSTM	GRU
With pre-processing	89	91,12	90
Without pre-processing	98,93	99,26	99,02

According to Table VI, for the problem of user identification, text preprocessing also reduces the identification accuracy, however, in contrast to the problem of determining hidden information leakage, the accuracy decreases not so critical. This allows us to conclude that the set of features for these two tasks is different.

All the further experiments are performed without preprocessing. Text preprocessing not only significantly affects the accuracy of the experiments, but also consumes computation time.

C. Analysis of various features extracting methods accuracy

We have performed experiments to estimate the accuracy of hidden leakages detection with the use of various feature extraction methods: manual feature extraction (listed in Table I), dictionary coding, and 4-grams. The experimental results are shown in Table VII.

TABLE VII. COMPARISON OF FEATURE EXTRACTION METHODS

Method	Accuracy,%			
	RNN	LSTM	GRU	CNN
Manual feature extraction	16	21	19	18
Dictionary coding	37	45	44	42
Character 4-gram	73	87	85	80

Experiments results showed that the manual feature extraction method and dictionary coding have the lowest accuracy. This means that the selected most informative features are not applicable to identify hidden information leaks. Dictionary coding also showed low accuracy since it does not consider spaces and character substitutions.

The best result in this experiment shows character 4-grams. N-grams allow you to define dependencies up to N characters. This means that every letter, extra space, or unknown character will be counted. N-grams allow you to consider the text from the point of view of each character combination separately.

It should also be noted that LSTM networks show the best results compared to GRU and RNN. This is largely due to the internal structure of the networks. Since n-grams showed the best accuracy, we will use them in a future experiment.

D. Study influence n-gram size on hidden leakages detection accuracy, GPU time, and memory consumption

A series of experiments were carried out to estimate the dependence processor time and memory consumption during training, and hidden leakages detection accuracy from n-gram size. Results are shown in Fig. 5 - 7.

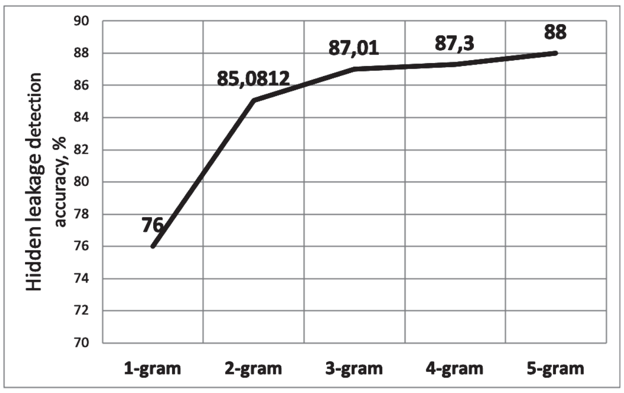

Fig. 5. Hidden leakages detection accuracy with n-grams of various size

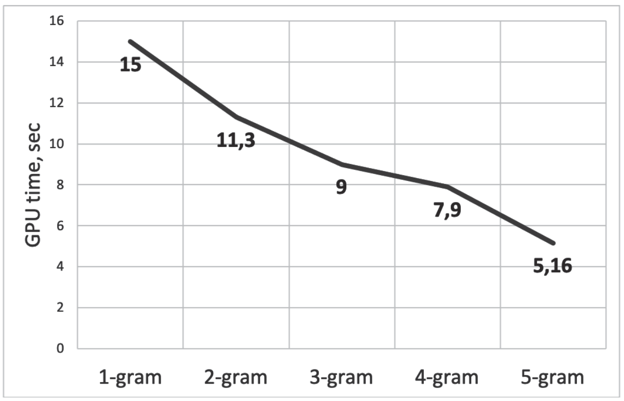

Fig. 6. GPU time on training stage with n-grams of various size

All experiments were carried out within the same session. As shown in the figures above, with an increase of n-grams size the accuracy increases. Since with an increase of n-grams size, the feature space dimension also enlarges, and the separability of classes increases with an increase in the dimension of the feature space. But memory consumption also grows significantly.

According to the results of the study, the most appropriate way to extract features is n-grams. This method allows you to extract the user's linguistic characteristics used in messages and identify their changes with the greatest accuracy.

With a relatively small increase in hidden leakages detection accuracy between 3-grams and 5-grams, the memory

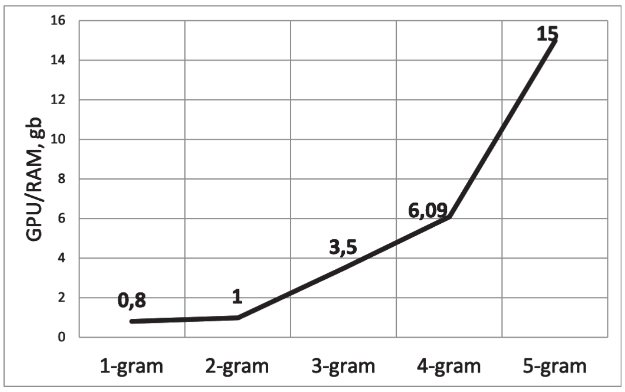

Fig. 7. Memory consumption on training stage with n-grams of various size

consumption increased up to tenfold. It is proposed to use character 4-grams. The average accuracy is 87%. In hiding 190 bits of information per 300-character text, the leak detection accuracy reaches up to 100%.

V. Conclusion

Steganography, like any other instrument, has two sides. It can be used legally and also it can be used as a tool for committing crimes. In this work, we have proposed a steganalysis approach allowing detection use of the text steganography methods based on authorship attribution or linguistic user identification methods. This approach allows to detect hidden leakages while detecting changes in the user's writing style using LSTM and 4-grams. Also, the framework for hidden leakages detection was developed. Experiments showed that the developed framework improves the accuracy of detecting hidden, implicit information leaks. The average accuracy while detecting hidden leaks in mixed mode reaches up to 87%. This tool is useful both for preventing information leaks and can assist in the hidden leakage investigation.

The impact of text preprocessing on the accuracy of identifying the use of steganography methods was assessed. The use of text preprocessing does not allow the use of many of the basic linguistic characteristics. In mixed mode, the detection accuracy decreased by 20%. It is recommended not to preprocess the texts.

In experiments, the LSTM networks showed the best accuracy (comparing with RNN, CNN, and GRU). This architecture is actively used for tasks related to natural language processing due to its structure, which allows you to remember previous states.

The dependence of the performance and identification accuracy for 1-gram, 2-gram, 3-gram, 4-gram, and 5-gram was studied. With an increase in the dimension of n-grams, there is an increase in the accuracy of detecting hidden leaks, as well as a decrease in processor time for operating, but this also increases memory consumption. Any algorithm needs a state that allows it to achieve a high level of accuracy with the most optimal resource consumption. As shown in the results of the experiment, the most effective way is to use 3-grams. When the bit depth is increased to 5 grams, the leak detection accuracy increases by only one percent, while the consumed memory is tenfold.

References

[1] A.A. Vorobeva, “Influence of Features Discretization on Accuracy of Random Forest Classifier for Web User Identification”, *Proceedings of the 20th Conference of Open Innovations Association FRUCT*, pp. 498-504, 2017.

[2] L. A. Kothari, K. Lipi, T. Rikin “Survey on Techniques of Data hiding on Web”, *International journal of innovative research in technology*, Volume 9 – 2016.

[3] Y.J. Bao, ”Text Steganalysis with Attentional L STM-CNN”, *2020 5th International Conference on Computer and Communication Systems (ICCCS)*, IEEE pp. 138-142, 2020.

[4] M. Chaumont, “Deep learning in steganography and steganalysis”, *Digital Media Steganography. – Academic Press*, pp. 321-349, 2020.

[5] M. Salomon, “Steganalysis via a convolutional neural network using large convolution filters for embedding process with same stego key: A deep learning approach for telemedicine”, *European Research in Telemedicine*, pp. 79-92, 2017.

[6] M. Agarwal, “Text steganography approaches: a comparison”, *International Journal of Network Security Its Applications (IJNSA)*, vol. 5, Jan 2013.

[7] I. Avcibas, N. Memon, B. Sankur “Steganalysis using image quality metrics”, *IEEE Transactions on Image Processing*, vol. 12, pp. 221-229, 2003.

[8] R. Chandramouli, “A mathematical framework for active steganalysis”, *ACM Multimedia Systems*, vol. 9, no. 3, pp. 303-311, 2003.

[9] J. J. Harmsen, W. A. Pearlman, “Steganalysis of additive noise modelable information hiding”, *Proceedings of the SPIE, Security, Steganography, and Watermarking of Multimedia Contents VI*, pp. 131-142, Jan 2003.

[10] K. Sullivan, U.Madhow, S.Chandrasekaran, B. Manjunath, “Steganalysis for Markov cover data with applications to images”, *EEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 275-287, 2006.

[11] Z. Yang, K. Wang, J. Li, Y. Huang, Y. Zhang, “TS-RNN: Text Steganalysis Based on Recurrent Neural Networks”, *IEEE Signal Processing Letters*, vol. 26, Dec 2019.

[12] P. Meng, L. Hang, W. Yang, Z. Chen, H. Zheng, “Linguistic Steganography Detection Algorithm Using Statistical Language Model”, *International Conference on Information Technology and Computer Science*, 2009.

[13] A.A. Vorobeva, ”Otbor informativnykh priznakov dlya identifikatsii Internet-pol’zovateley po korotkim elektronnym soobshcheniyam”, *Nauchno-tekhnicheskiy vestnik informatsionnykh tekhnologiy, mekhaniki i optiki*, pp. 117–12, 2017.

[14] Y.V. Dmitrin, D.S. Botov, J.D. Klenin, I.E. Nikolaev, ”Comparison of deep neural network architectures for authorship attribution of russian social media texts”, *Komp’juternaja lingvistika i intellektual’nye tehnologi*, 2018.

[15] H. J. Escalante, T. Solorio, “Local histograms of character n-grams for authorship attribution”, *In Annual Meeting of the Association for Computational Linguistics: Human Language Technologie*, pp. 288–298, 2011.

[16] U. Sapkota, S. Bethard, M. Montes-y Gomez, T. Solorio, “Not all character n- grams are created equal: A study in authorship attribution.”, *In Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, pp. 93–102, 2015.

[17] M. Popescu, C. Grozea, “Kernel methods and string kernels for authorship analysis”, *LEF 2012 Evaluation Labs*, 2012.

[18] A. Bartoli, A. Dagri, A. D. Lorenzo, E. Medvet, F. Tarlao, “An author verification approach based on differential features”, *CLEF 2015 Evaluation Labs*, 2015.

[19] I. Rahul, “A Machine Learning Framework for Authorship Identification from Texts”, *Language Technologies Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA*, Aug 2019.

[20] T. Litvinova, O. Litvinlova, O. Zagorovskaya, P. Seredin, A. Sboev, “Ruspersonality: A Russian corpus for authorship profiling and deception detection”, *FRUCT Conference on Intelligence, Social Media and Web (ISMW FRUCT)*, pp. 1–7, 2016.

[21] *A*.A. Vorobeva, “Examining the Performance of Classification Algorithms for Imbalanced Data Sets in Web Author Identification”, *Proceedings of the 18th Conference of Open Innovations Association FRUCT*, pp. 385-390, 2016.

[22] *A*.A. Khazagarov, A.A. Vorobeva, “Protivodeystviye skrytym utechkam informatsii s pomoshch’yu metodov lingvisticheskoy identifikatsii”, *Kulaginskiye chteniya: tekhnika i tekhnologiya proizvodstvennykh protsessov: sbornik statey*, vol. 1, pp. 44-47, 2019.

[23] *G*. Margarov, “Investigation of Web Based Hidden Dara”, *Web Intelligence and security*, 2010.

[24] *H*. Huanhuan, Z. Xin, “Adaptive Text Steganography by Exploring Statistical and Linguistical Distortion”, *2017 IEEE Second International Conference on Data Science in Cyberspace (DSC)*, Aug 2017.

[25] *Z*. Yang, “A Fast and Efficient Text Steganalysis Method”, *IEEE Signal Processing Letters*, vol. 26, Apr 2019.

[26] *A*. Naharuddin, A. Dharma, S. Sumpeno, “A High Capacity and Imperceptible Text Steganography Using Binary Digit Mapping on ASCII Characters”, *2018 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, May 2019.