computing@computingonline.net
www.computingonline.net

Print ISSN 1727-6209
On-line ISSN 2312-5381
International Journal of Computing

INTRUSION DETECTION IN COMPUTER NETWORKS USING LATENT SPACE REPRESENTATION AND MACHINE LEARNING

Vladyslav Hamolia [1)], Viktor Melnyk [1,2)], Pavlo Zhezhnych [3)], Anna Shilinh [4)]

1) Department of Information Technologies Security, Lviv Polytechnic National University, 12 S. Bandery str., Lviv, 79000, Lviv, Ukraine
2) Institute of Mathematics, IT and Landscape Architecture, John Paul II Catholic University of Lublin, Konstantynow 1H, Lublin, Poland, viktor.melnyk@kul.pl
3) Department of Social Communication and Information Activities, Lviv Polytechnic National University, 12 S. Bandery str., Lviv, 79000, Ukraine, pavlo.i.zhezhnych@lpnu.ua
4) Department of Information Systems and Networks, Lviv Polytechnic National University, 12 S. Bandery str., Lviv, 79000, Lviv, Ukraine, AnnaShiling@gmail.com

Paper history:
Received 12 May 2020
Received in revised form 27 June 2020
Accepted 08 July 2020
Available online 27 September 2020

Keywords:
intrusion detection;
machine learning;
clustering;
traffic detection;
anomalies;
neural nets.

Abstract: Anomaly detection (AD) identifies samples that are not related to the overall distribution in the feature space. This problem has a long history of research through diverse methods, including statistical and modern Deep Neural Networks (DNN) methods. Non-trivial tasks such as covering ambiguous user actions and the complexity of standard algorithms challenged researchers. This article discusses the results of introducing an intrusion detection system using a machine learning (ML) approach. We compared these results with the characteristics of the most common existing rule-based Snort system. Signature Based Intrusion Detection System (SBIDS) has critical limitations well observed in a large number of previous studies. The crucial disadvantage is the limited variety of the same attack type due to the predetermination of all the rules. DNN solves this problem with long short-term memory (LSTM). However, requiring the amount of data and resources for training, this solution is not suitable for a real-world system. This necessitated a compromise solution based on DNN and latent space techniques.

1. INTRODUCTION

The basic concept for anomaly detection (AD) is monitoring regular and irregular network behavior. AD has several complications that are not typical of traditional machine learning. Firstly, anomalies are significantly fewer than nominal patterns. Secondly, there is no rigid boundary distributing anomalies and nominals.

The most common approach to network security is utilizing several techniques for protection, route tracking, authentication, etc. Complex Intrusion detection systems (IDS) have an input and output predefined in advance. It is a brand-new level able to expand or substitute other mechanisms in the network security architecture.

According to the existing study [1], there are difficulties in SBIDS like handling the unknown attack and LSTM's problems such as computational expense and the tendency to overfitting. This research aims at finding an alternative solution without these difficulties by comparing the *NSL-KDD* dataset of our ML approach with the popular *Snort* [2] open-source IDS and the LSTM architecture. In synthetic tests, the empirical results confirmed the better performance of the proposed approach using Deep Neural Networks (DNN) output as a latent space in combination with the one-class Support Vector Machine (SVM) classification [3] method.

2. ANOMALY DETECTION TECHNIQUES

The IDS can implement a specific technique or a set of methods such as signature analysis, traffic monitoring, anomaly detection, etc.

The *Signature-based IDS* (SBIDS) belongs to the attack detection searching for specific patterns, such as byte sequences in network traffic or known malicious intrusion sequences used by malware. This terminology originates from anti-virus software referring to these detected patterns as signatures. Therefore, it can only identify an attack if there is an accurate matching behavior against the stored or known patterns termed as signatures.

The *Anomaly-based IDS* (ABIDS) detects unknown attacks due to the rapid development of malware. Their basic approach is using ML to create a model and extract nonlinear features of the trustworthy activity, as well as to compare this model with new behavior in the network. Since these models are amenable to training according to the applications and hardware configurations, the ML-based method has a better generalizing property observed in [4] and [6] in comparison to traditional signature-based IDS. Most attempts to build ABIDS are conceptual models aiming at testing the possibility of applying mathematical modeling.

Generally, all methods [7] designed for the detection of anomalies form such groups:

– based on the storage of examples of behavior;
– based on frequency distribution and Bayesian Networks;
– modeling anomalies detection using ML models (including DNN).

It is possible to combine all of these approaches. For example, the frequency analysis is suitable for post-processing of the ML results; through the signatures, it is achievable to detect the most trivial cases of anomalies ahead of the entire ABIDS system. However, some combinations can be more efficient, for example, feature extraction by DNN and the following use of these features as an input to ML algorithms.

3. MACHINE LEARNING IN ANOMALY DETECTION

Applying machine learning techniques, we can automatically construct a model based on the training data set containing records of individual observations. It is possible to describe records employing a set of attributes (features) and associated labels. A typical IDS pipeline, which includes machine learning, has the following stages:

– monitored environment exploration,
– feature engineering (FE) – the process of extraction of the most essential attributes from the raw data,
– ML model training,
– detection of an anomaly,
– intrusion report.

Various machine learning techniques have been used to develop IDS such as DNN, Support Vector Machines (SVM), Naive-Bayesian (NB), Self-Organizing Maps (SOM), K-Nearest Neighbors (KNN), and Decision Tree (DT). All these ML techniques are trained in a supervised or unsupervised manner to identify the normal and attack packets in network traffic. With the increase of network bandwidth and traffic speed, the difficulties with traditional ABIDS are packet loss, slow detection, and higher response time to deal with the massive network data.

The algorithms differ in their approaches to solving the AD problem. However, at an abstract level, all of them attempt to create a decision boundary – the plane in multidimensional space to split into two entities (normal and attacks), as in the synthetic example in Fig. 1:

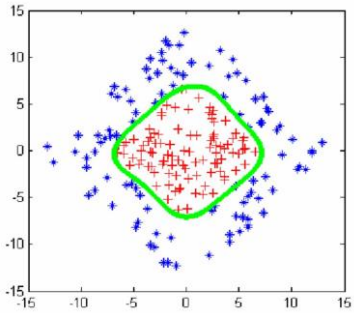

Figure 1 – Principle of normal and anomaly data distinguishing

In the paper, we are going to observe how the decision boundary, created by the ML method, separates the anomaly entities from normal ones.

4. DATA USED FOR ABIDS EVALUATION

To verify the ABIDS comparison hypothesis, we have chosen the NSL-KDD dataset as an input to selected models. The dataset has 41 attributes unfolding various features of the traffic flow. A label is assigned to each of them either as a particular attack type or as a normal one. The details of the attributes, namely their names, description, and sample data, are given in [9]. Table 1 and Table 2 present the example of attack classes (which our final model will attempt to predict) and attack types based on our previous exploration data analysis.

To validate our model and build our vision of its reliability, we divided the data into two types: training and testing. We conducted this separation using *stratified sampling*, which means creating two groups of data based on the target variable (whether the record refers to an anomaly or a regular sampling space).

Table 1. Attack classes and examples of different attack types

Attack Class	Attack Types
DoS	Back, Land Neptune, Pod, Smurf
Probe	Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint
R2L	Guess password, Ftp write, Imap
U2R	Buffer overflow, Loadmodule

Table 2. Distribution of dataset based on attack type

Data type	Amount of data samples					
	All	Norm	DoS	Probe	R2L	U2R
Train	125973	67343	45927	11656	995	52
	%	53.46	36.45	9.25	0.79	0.04
Test	22543	9711	7458	2421	2754	200
	%	43.08	33.08	10.7	12.2	0.9

We will use the following data to train and evaluate selected ABIDS. More analysis of dataset can be found in [9].

5. COMPARISON OF THE MACHINE LEARNING ALGORITHMS USED IN ABIDS

Numerous unsupervised methods were applied to solve the problem of detecting anomalies and improving ABIDS rates at all levels, such as clustering, factor analysis, etc. Based on the description of various unsupervised anomaly detection algorithms, Table 3 shows a comparison of the most common algorithms, taking into account the specifications and the mathematical background of each of them.

Table 3: Pros and cons of ML algorithms

Technique	Pros	Cons
KNN	1. Computationally cheap. 2. No data size restriction. 3. Low complexity.	1. Heavy to store all results. 2. Requires domain knowledge for feature extraction. 3. Cannot handle difficult dependencies.
DNN	1. Automatic feature extraction. 2. Modeling of non-linear dependencies. 3. Various architectures for supervised and unsupervised tasks. 4. Transfer learning.	1. Computationally expensive (based on the size of DNN). 2. Huge amount of data for training.
SVM	1. Use kernel trick to detect dependencies. 2. Efficient in cases of high-dimensional data.	1. Requires both positive and negative examples. 2. Results of kernel function is not easy to interpret.
DT	1. Easy to interpret. 2. Requires little data preparation. 3. Able to handle both numerical and categorical data.	1. DT learners create over-complex trees that do not generalize the data well. 2. There are concepts that are hard to learn because decision trees do not express them easily.

6. IMPLEMENTED SOLUTION

Some of the ML methods under consideration have non-interceptable issues that differ in nature and complexity illustrated in [8] and [9]. We focused mainly on FE and computational tasks. To build a robust automatic IDS, we selected *Fully Connected* DNN (FCDNN) as the FE part of the general ML flow (mentioned in Section 3). Typically, this part is performed manually using the previous domain analysis, data personality, etc.

In FCDNN, each neuron in one layer is connected to all neurons in the next layer. Such an architecture allows gaining performance in various ML tasks, but it tends to overfit. However, it can be used to embed input [10] data and represent a record in a latent space [11]. Latent space is a representation of squashed data, which form a new space where similar items have small distance.

As a classifier (performing an attack classification), the FCDNN hidden unit outputs operate as an implementation of a nonlinear projection of high-dimensional input (features) space onto a lower and denser (abstract) feature space. In this space, we outlined records for better separation using the network output layer. Furthermore, the visualization of the latest hidden internal representations may facilitate the identification of data structures. With this approach, the classifier ideally acts as an FE.

Although feature extraction training is not a classifier, it is based on class label information and is therefore supervised. The number of input units (Fig. 2) is specified as the number of objects, and the number of output units is specified as pattern classes.

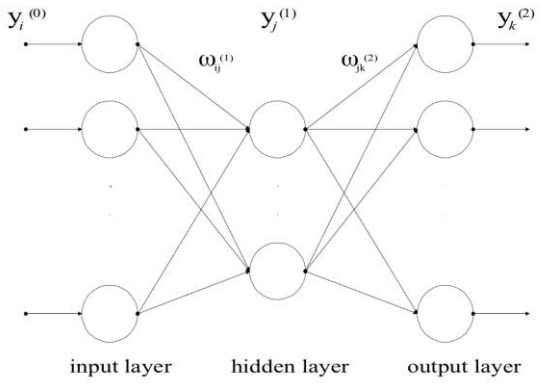

Figure 2 – Example of FCDNN for feature extraction using 2 layers (the *hidden layer* is used as a latent space, the output of each layer marked as *Yi.* and the weights between layers marked as w_j)

Following the task, we designed and selected a set of hidden layers (or a *backbone*) for exploratory data projection, classification, etc. We use 5 layers of the following sizes: 30, 24, 20, 18, 12, to classify

the type of attack for the training data set. The last layer with 12 neurons will be used as a *latent space*. The target has 6 classes: 5 classes of attack types and 1 class for innocuous records.

Another problem, commonly associated with DNN, is a requirement of extra computational resources. In our solution, we separate the FE (using *Fully Connected* DNN) from DNN and apply the classification of the one-class SVM with the RBF kernel (which shows the best result locally and it has important property, it is invariant to transition) instead of the part of the DNN classification. One-class SVM attempts to find decision boundaries by mapping the nominal data to high-dimensional kernel space and separating them from the source with maximum margin. Various techniques were observed in [5] and [6].

We introduced slack variables ξ to prevent the SVM classifier from overfitting with noisy data (or to create a soft margin [7]). They allow some data points to lie within the margin. The constant C > 0 determines compromise between maximizing the margin size and the number of training data points within that margin (and training errors) to maximize the margin.

SVM has following minimization expression:

$$\min_{w,b,\xi_i}\left[\frac{\|w\|^2}{2}+C\sum_{i=1}^{n}\xi_i\right] \tag{1}$$

subject to:

$$y_i(w^T\phi(x_i)+b)\geq 1-\xi_i$$
$$\xi_i \geq 0 \quad \text{for all } i = 1, \ldots, n$$

Here ξ is used as a slack variable to add an inequality constraint, to transform it into equality, or to ease constraints.

For our experimental setup, we chose Tensorflow 1.14 as the main library for setting up the model and configured our DNN (for 5 classes of attack types) with the following hyperparameters (fine tuned with approach described in [19]):

- Adam optimizer (which performs the best according to our experiments) [12]:
  - learning_rate=0.0005 (with an *exponential decay*);
  - beta1=0.85;
  - epsilon=1e-07;
- batch size: 32;
- epoch: 10;
- cross validation [13], [14]: 5 folds.

We have used SVM classifier from sklearn 0.21.3 with following hyperparameters:

- C=1.2;
- kernel=rbf [15];
- degree=5.
- Testing hardware:
  - Intel core i5-7300HQ
  - RAM: 8GB DDR4, 2400Mhz

With given hyperparameters, it will be possible to train models concerning the objective function of models and reproduce our solution.

7. EVALUATION AND COMPARISON OF OUR SYSTEM WITH EXISTING IDS SOLUTIONS

We evaluated the effectiveness through training and testing the NSL-KDD datasets discussed in Section 4. We used the following entities to evaluate: True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). We used these entities to compute the following various indicators:

- *Accuracy* (A), which is defined as the percentage of correctly classified records in their total number.
- *Precision* (P), which is referred as Positive Predictive Value (PPV), defined as the % ratio of the number of TP records divided by the sum of TP and FP classified records.
- *Recall* – referred to the TP rate (or sensitivity) and defined as the % ratio of the number of TP records divided by the sum of TP and FN classified records.
- *F-Measure* – a measure to represent test accuracy, defined as the harmonic mean of precision and recall, which represents a balance between them.

We assume that our model has found the best solution, and is consistent with the training data at the training stage. We select an independent sample of verification data from the population sample as training data. It generally turns out that most models tend to overfit: there is a huge gap between results on the testing and training samples. Many methods for constructing the right validation strategies [17] allow us to expect that the model will have evaluation results on unseen data, as well as on test data. Cross-validation attempts to estimate this difference.

To test our model for robustness, we use an approach called *stratified k-fold cross-validation* [18] shown in Fig. 3. For each fold (one split of dataset), we randomly remove 2 types of attacks from different attack classes of the training dataset and fit the model (Run 1-4 in Figure 3). Yet, we leave them in the test dataset for examining the model's ability to acquire some general concepts and not to overfit the training data [19].

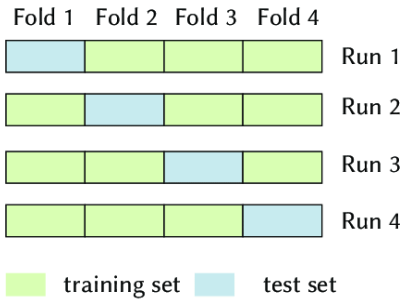

Figure 3 – Example of stratified k-fold validation

In our experiment, we made 5 folds and averaged the results of each fold. The accuracy metric should be evaluated separately for individual attack classes according to the nature of the evaluation metric.

Table 4 shows the accuracy metric for individual classification:

Table 4. Comparison results of proposed system with the *Snort* system

Attack type / Metric (%)	Normal		DoS		Probe	
	Snort	DNN+ SVM	Snort	DNN+ SVM	Snort	DNN+ SVM
Accuracy	73.9	95.1	75.4	94.9	67.1	87.1
Precision	75.1	92.8	78.1	95.7	68.1	84.5
Recall	68.1	92.6	79.1	97.1	58.1	85.3
F-Measure	71.4	92.7	78.6	96.4	62.7	84.9

The presented solution is more lightweight in contrast to most of DNN solutions. However, we tried to keep the results score without sacrificing performance. We have already mentioned one of the most popular approaches using LSTM [1] networks. In Table 5, we compare our model with the one provided in [2], which deploys the LSTM network.

Table 5. Comparison results of proposed system with LSTM system

Approach / Metric	DNN +SVM	LSTM
Accuracy (%)	93.2	94.5
Precision (%)	91.2	93.7
Recall (%)	89.8	91.7
F-Measure (%)	90.4	92.7
Train time (sec)	138	891
Time to predict 1K records (sec)	0.03	0.13

As we can see in Table 5, the LSTM model gives better results in the test. It is time-consuming to retrain it, though. Such criteria might be crucial in real-world tasks [21] [22], and one should select the type of system based on personal intentions.

8. CONCLUSION

In this paper, we examined the traditional approaches to anomalies detection, namely ML-based and SBIDS methods. ML techniques were under consideration of the intrusion detection researchers to eliminate the deficiencies of knowledge base detection techniques. Also, our results display a tremendous difference in performance between our model and the models we analyzed.

Evaluation experiments and the results of various metrics confirmed that the proposed solution deals with the main difficulties considered in the article: it solves crucial problems of SBIDS, such as handling unknown attacks, and LSTM`s problems, such as computational expense and the tendency to overfit.

The proposed method, based on latent space, provides a reduced number of features (the final layer has output which contains 12 neurons) and improves the detection accuracy of multiple attack classes. The conducted research demonstrates that the approaches using machine learning techniques provide better results for classification tasks. A proper dataset with a sufficient quantity of samples should be developed for individual attack classes to better training and proper feature extraction. Training of each hidden layer will yield in the better feature selection process, but it takes significant time.

There are a lot of DNN architecture solutions that should be validated with the proposed validation method and a latent space representation. We also plan to build some mechanism for interpretation of developed approach, and get better visibility of training and evaluation process.

9. REFERENCES

[1] S. Merity, N. S. Keskar, and R. Socher, *Regularizing and Optimizing LSTM Language Models*, arXiv preprint arXiv:1708.02182, 2017, [Online]. Available at: https://arxiv.org/abs/1708.02182

[2] V. Kumar and O. P. Sangwan, "Signature base intrusion detection system using SNORT," *International Journal of Computer Application & Information Technology*, vol. 1, no. 3, pp. 35-41, 2012.

[3] T. Joachims, "Making large-scale SVM learning practical," In B. Scholkopf, C.J.C. Burges, and A. J. Smola (Eds.), *Advances in Kernel Methods – Support Vector Learning*, Cambridge, MA: MIT Press, pp. 169-184, 1999.

[4] S. Lundberg and S. Lee, *A Unified Approach to Interpreting Model Predictions*, arXiv preprint

arXiv:1705.07874, 2017, [Online]. Available at: https://arxiv.org/pdf/1705.07874.pdf.

[5] S.S. Khan, M.G. Madden, "One-class classification: Taxonomy of study and review of techniques", *The Knowledge Engineering Review*, no. 29 (03), pp. 345–374, 2014.

[6] W. Zhu, P. Zhong, "A new one-class SVM based on hidden information," *Knowledge-Based Systems*, no. 60, pp. 35–43, 2014.

[7] B. J. Radford, L. M. Apolonio, A. J. Trias, and J. A. Simpson, *Network Traffic Anomaly Detection Using Recurrent Neural Networks*, arXiv preprint arXiv:1803.10769, 2018, [Online]. Available at: http://arxiv.org/abs/1803.10769

[8] R. Guidotti, A. Monreale, F. Turini, D. Pedreschi, and F. Giannotti, *A Survey of Methods for Explaining Black Box Models*, arXiv preprint arXiv:1802.01933, 2018, [Online]. Available at: https://arxiv.org/pdf/1802.01933.pdf.

[9] L. Dhanabal, D. S. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, no. 6, pp. 446-452, 2015.

[10] J. Cha, K. S. Kim, S. Lee, *On the Transformation of Latent Space in Autoencoders*, arXiv preprint arXiv:1901.08479, 2018, [Online]. Available at: https://arxiv.org/abs/1901.08479

[11] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, Dec. 2010, [Online]. Available at: http://www.jmlr.org/papers/volume11/vincent10a/vincent10a.pdf

[12] D. P. Kingma, J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv preprint arXiv:1412.6980, 2015, [Online]. Available at: https://arxiv.org/pdf/1412.6980.pdf.

[13] Y. Bengio and Y. Grandvalet, "Bias in estimating the variance of K-Fold cross-validation," *Statistical Modeling and Analysis for Complex Data Problems*, vol. 1, pp. 75-95, 2005.

[14] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*, Wiley, 2004, 350 p.

[15] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Machine Learning*, vol. 46, pp. 131–159, 2002.

[16] Liu H. and Yu L., "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, issue 4, pp. 491–502, 2005.

[17] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, issues 5-6, pp. 602-610, 2005.

[18] W. Siblini, J. Fréry, L. He-Guelton, F. Oblé, Y.-Q. Wang, *Master your Metrics with Calibration*, arXiv preprint arXiv:1909.02827, 2019, [Online]. Available at: https://arxiv.org/pdf/1909.02827.pdf.

[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. [Online]. Available at: http://dx.doi.org/10.1038/nature14539.

[20] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, pp. 2295–2329, 2017.

[21] K. Kawaguchi, "Deep learning without poor local minima," *Advances in Neural Information Processing Systems*, vol. 29, pp. 586–594, 2016.

[22] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams et al., "Recent advances in deep learning for speech research at Microsoft," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, pp. 734-748 2013.

[23] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through FFTs," *Proceedings of the International Conference on Learning Representations (ICLR2014)*, Banff, Canada, 2014. [Online]. Available at: https://arxiv.org/abs/1312.5851

[24] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al., *Tensorflow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*, arXiv preprint arXiv:1603.04467, 2016, [Online]. Available at: https://arxiv.org/pdf/1603.04467.pdf

[25] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in Neural Information Processing Systems*, pp. 1135–1143, 2015.

Vladyslav Hamolia *is a PhD student at the department of Information Technologies Security at Lviv Polytechnic National University. He is specialized in performing data analysis and provides support in making complex business decisions based on it, building predictive models, performing data preprocessing and exploratory data analysis. His research areas mostly lie in statistical learning, predictive analytics, NLP, signal processing, anomaly detection. He has experience in development of computer vision systems, analyzing and detecting patterns, implementing algorithms in order to integrate business logic of large companies.*

Viktor A. Melnyk *is a professor of the Department of Information Technologies Security at Lviv Polytechnic National University. He was awarded with the academic degrees of Philosophy Doctor in 2004, and Doctor of Technical Sciences in 2013 at Lviv Polytechnic National University. He has scientific, academic and hands-on experience in the field of computer systems research and design, proven contribution into IP Cores design methodology and high-performance reconfigurable computer systems design methodology. He is experienced in computer data protection, including cryptographic algorithms, cryptographic processors design and implementation, wireless sensor network security.*

Pavlo I. Zhezhnych *is a Professor, Sc.D, professor at the department of Social Communication and Information Science, Institute of Humanities and Social Sciences, Lviv Polytechnic National University. In 1996 he graduated from the Faculty of Applied Mathematics, at Lviv Franko State University, the specialty "Applied mathematics". In 2001 he defended his thesis for the degree of Candidate of technical sciences in Lviv Polytechnic National University. In 2009 he defended his thesis for the degree of doctor of technical sciences. His Sc.D. thesis title is "Methods and means of relational time-dependent databases organizing". The scientific and professional interests are focused in the fields of relational and temporal database modeling, Data analysis, Data mining, Information security, Web technologies.*

Anna Yu. Shilinh *is a Ph.D., assistant lecturer at the department of Social Communication and Information Science at Lviv Polytechnic National University. She graduated from Drogobych State Pedagogical University, specialty "Informatics. Applied Mathematics". Her fields of scientific activity are internet linguistics, research on informational activities in the web space.*