STEGANALYSIS OF PQ ALGORITHM BY MEANS OF NEURAL NETWORKS

Zuzana Oplatkova
Tomas Bata University in Zlín
Faculty of Applied Informatics
Department of Informatics and Artificial Intelligence
Nam. T.G. Masaryka 5555
760 05, Zlín, Czech Republic
E-mail: holoska@fai.utb.cz

Jiri Holoska
Tomas Bata University in Zlín
Faculty of Applied Informatics
Department of Informatics and Artificial Intelligence
Nam. T.G. Masaryka 5555
760 05, Zlín, Czech Republic
E-mail: oplatkova@fai.utb.cz

Ivan Zelinka
VB-TUO
Faculty of Electrical Engineering and Computer Science
Department of Computer Science
17. listopadu 15
708 33 Ostrava-Poruba, Czech Republic
E-mail: ivan.zelinka@vsb.cz

Roman Senkerik
Tomas Bata University in Zlín
Faculty of Applied Informatics
Department of Informatics and Artificial Intelligence
Nam. T.G. Masaryka 5555
760 05, Zlín, Czech Republic
E-mail: senkerik@fai.utb.cz

Roman Jasek
Tomas Bata University in Zlín
Faculty of Applied Informatics
Department of Informatics and Artificial Intelligence
Nam. T.G. Masaryka 5555
760 05, Zlín, Czech Republic
E-mail: jasek@fai.utb.cz

KEYWORDS

Steganalysis, Neural networks, Huffmann coding.

ABSTRACT

The paper deals with a steganalysis of PQ algorithm by means of neural networks. The paper continues with the research of steganalysis by means of neural networks and brings results for the other steganography tool and also simulation results for different settings of neural network.

INTRODUCTION

Our previous research on classification of stego and cover images by means of ANN was introduced in (Oplatkova 2008a, Oplatkova 2008b, Oplatkova 2009) where detection of 3 stego programmes were presented - OutGuess (www.outguess.org), Steghide (Hetzl 2008), CipherAWT with F5 algorithm (Fridrich 2002). The number of programmes for inserting stego content is increasing and then the research in the steganalysis of further tools is required. This paper introduces a detection of PQ algorithm (Fridrich 2004).

Steganalysis is connected with information security. Mainly in companies, information security is a very spoken problem nowadays. All employers have to pay attention to their employees if company secrets and know-how are not spread out of the company. One of the possibilities how to leak the information is to use a steganography (Cole 2003). Steganography (Cole 2003) and cryptography (Goldwasser 2001) are connected together more or less. Cryptography is strong in the usage of the key and the message is somehow coded. But if it is sent unsecure, attacker will notice it very soon and will try to break it. Therefore steganography helps with secure transfer of secret messages. It codes a message inside the picture or other multimedia files which can be sent e.g. via emails. If you see such a picture, normally you do not recognize that there is a secret message. And this is the point. Crackers will go through and will not give the attention to the message.

Therefore to have a detector of steganography content in the multimedia files is very important. To reveal a steganography content is called steganalysis, i.e. a detection of files with hidden information of without hidden information which was inserted by means of steganography.

The PQ algorithm differs from the previous used stego tools so that in the training set greyscale images are used instead of full coloured ones. The same mean for extraction of information was applied as in previous research – Huffman coding. It extracts 64 parameters with numerical values as artificial neural networks need numerical values in the input layer for their run.

Firstly, PQ algorithm is mentioned, next paragraph will continue with information about Huffman coding and after that artificial neural network used in simulations are described. Results and conclusion follow.

PQ ALGORITHM

Perturbed quantization (PQ) steganography (Fridrich 2004, Goldwasser 2001, Hetzl 2008) is a quite successful data hiding approach for which current steganalysis methods fail to work (http://aminet.net/package/util/crypt/jstegsrc). In other words, PQ does not leave any traces in the form that the current steganalysis methods can catch. However, linear dependency between image rows and/or columns in the

Proceedings 25th European Conference on Modelling and Simulation ©ECMS Tadeusz Burczynski, Joanna Kolodziej Aleksander Byrski, Marco Carvalho (Editors)
ISBN: 978-0-9564944-2-9 / ISBN: 978-0-9564944-3-6 (CD)

spatial domain is affected by PQ embedding due to random modifications on discrete cosine transform (DCT) coefficients' parities during data hiding.
In PQ steganography, the cover object is applied an information reducing operation that involves quantization such as loss compression, resizing, or A/D conversion before data embedding. The quantization is perturbed according to a random key for data embedding, therefore called "perturbed quantization." PQ steganography, which uses JPEG compression for information reducing operation, is different from their DCT-based counterparts. Since message bits are encoded by changing DCT parities after quantization, the cover image can be thought of just as a recompressed input image. To achieve high embedding rates, recompression is realized by doubling the input quantization table with the assumption that recompression of cover JPEG images does not draw any suspicion because of its wide usage in digital photography. Since the original cover image is recompressed via embedding operation, its compressed version should be considered as "cover" instead of original image.

HUFFMAN CODING

Huffman coding was used to extract information from images as ANN needs numerical values for its run. Huffman coding was designed by David Huffman in 1952 [Cormen 2001]. This method takes symbols represented (e.g. by values of discrete cosine transformation as in our case, which is one of methods how to present information in pictures like colour, brightness etc.) and coded it into changeable length code so that according to statistics the shortest bit representation to symbols with the most often appearance. It has two very important properties – it is a code with minimal length and prefix code that means that it can be decoded uniquely. On the other hand, the disadvantage is that we must know appearance of each symbol a priori. But in the case of pictures we can work with estimation, which will be edited during the compression.
To demonstrate more how inserting of the hidden information works, following two pictures (Figure 1 a) and b)) can be used which visualize the Huffman coding. Each bit word can stand as a brick in the wall. It is possible to get two same big walls but each one will be assembled from different bricks and brick sizes. These two walls have the same size but of different structure (different set of bricks, some bricks appear more often then others). By same analogy, differences in cover and stego files can be viewed. The aim is to compare the different bit word length and different sizes of bricks in the walls for cover and images affected by steganography.

a)

b)

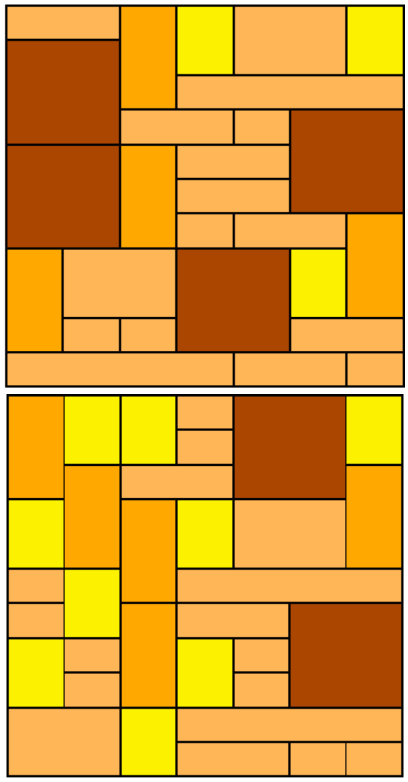

Figure 1: Illustration of Huffman coding histogram – a) cover image, b) stego image.

As the main goal of steganography is not to attract attention, stego images appear as usual pictures taken by digital camera. But there are significant changes in the structure of stego images. These changes in JPEG structure are relevant and used in this case for correct training of artificial neural network.

NEURAL NETWORKS

Artificial neural networks are inspired in the biological neural nets and are used for complex and difficult tasks. As in the case of this research, the most often usage is classification of objects. ANN are capable of generalization and hence the classification is natural for them. Some other possibilities are in pattern recognition, control, filtering of signals and also data approximation and others.
Simulations were performed with feedforward net with supervision. ANN needs a training set of known solutions to be learned on them. Supervised ANN has to have input and also required output. ANN with unsupervised learning exist and there a capability of selforganization is applied.
The neural network works so that suitable inputs in numbers have to be given on the input vector. These inputs are multiplied by weights which are adjusted during the training. In the neuron the sum of inputs multiplied by weights are transferred through mathematical function like sigmoid, linear, hyperbolic tangent etc. Therefore ANN can be used for data approximation (Hertz 1991, Freeman 1994).

These single neuron units (Figure 2) are connected to different structures to obtain ANN (e.g. Figure 3). These networks were design for different tasks.

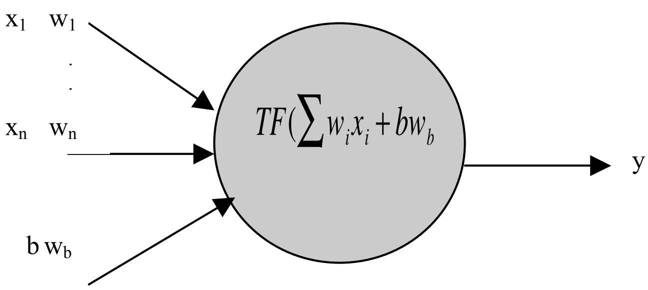

Figure 2: Neuron model, where TF (transfer function like sigmoid), x_1 - x_n (inputs to neural network), b – bias (usualy equal to 1), w_1 – w_n, w_b – weights, y – output.

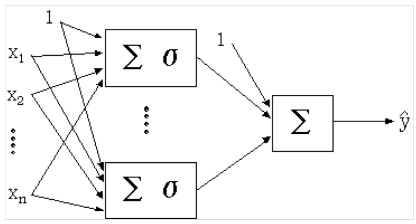

Figure 3: ANN models with one hidden layer, where $\sum \delta = TF[\sum(w_i x_i + bw_b)]$ and in this case $\sum = TF[\sum(w_i x_i + bw_b)]$, where TF is sigmoid. The picture is taken from Neural Networks Toolbox for Mathematica environment (www.wolfram.com) since this tool was used during the simulations. Also names are taken from this tool to avoid other speculations what it means..

Future simulations expect a usage of soft computing algorithms for optimization of suitable parameters or structure called evolutionary, e.g. Self-Organizing Migrating Algorithm (Zelinka, 2004), Differential Evolution (Price, 1999) or HC12 (Matousek, 2010). Also a different kinds of neural networks will be used, e.g. RBF nets with implementation of GAHC algorithm (modification of HC12) (Matousek, 2011) for design of structure and estimation of parameters.

RESULTS

During the simulations several cases have been tested – different settings for transfer functions in hidden neurons and output neuron and different number of hidden neurons.

The training set was prepared with following length of inserted messages: 5, 10, 15, 30, 75, 150, 300, 600 Bytes and used resolutions: original and changed to 800x600, 1024x768, 1280x1024, 1440x900, 1680x1050, 1920x1440, 2560x1600.

Some examples will follow:

PQ algorithm, 1 hidden neuron, saturated linear in hidden and output neurons.

Following (Figure 4) shows the training root mean square error (RMSE).

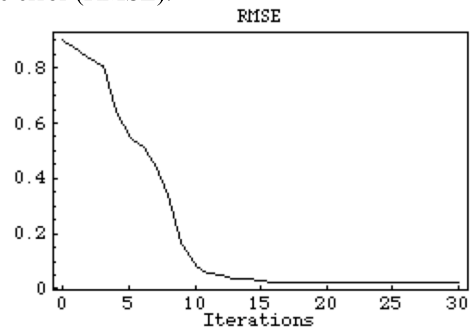

Figure 4: Example of training RMSE for 1 hidden neuron, saturated linear in hidden and output neurons

Following tables (Tab. 1-9) deals with resolutions without two highest because of the space, the results are similar as in other resolutions.

Table 1 - 9: Results for PQ algorithm, 1 hidden neuron, saturated linear in hidden and output neurons, different length of messages and 5 resolutions.

Message 5 bytes	**800 x 600**	**1024 x 768**	**1280 x 1024**	**1440 x 900**	**1680 x 1050**
Samples total	2081	2077	2076	2078	2083
Correct classificcaiton	2035	2007	1956	1993	2023
Missclassification	46	70	120	85	60
Success rate in %	97.79	96.63	94.22	95.91	97.12
Error rate in %	2.21	3.37	5.78	4.09	2.88

Message 10 bytes	**800 x 600**	**1024 x 768**	**1280 x 1024**	**1440 x 900**	**1680 x 1050**
Samples total	2081	2077	2076	2078	2083
Correct classificcaiton	2035	2007	1956	1994	2023
Missclassification	46	70	120	84	61
Success rate in %	97.79	96.63	94.22	95.95	97.07
Error rate in %	2.21	3.37	5.78	4.05	2.93

Message 15 bytes	**800 x 600**	**1024 x 768**	**1280 x 1024**	**1440 x 900**	**1680 x 1050**
Samples total	2081	2077	2076	2078	2083
Correct classificcaiton	2038	2008	1958	1995	2023
Missclassification	43	69	118	83	61
Success rate in %	97.93	96.68	94.32	96.01	97.07
Error rate in %	2.07	3.32	5.68	3.99	2.93

Message 30 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2081	2077	2076	2078	2083
Correct classificcaiton	2035	2008	1958	1993	2023
Missclassification	46	69	119	85	61
Success rate in %	97.79	96.68	94.28	95.91	97.07
Error rate in %	2.21	3.32	5.72	4.09	2.93

Message 75 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2081	2077	2076	2078	2083
Correct classificcaiton	2035	2008	1958	1993	2023
Missclassification	44	69	118	85	61
Success rate in %	97.88	96.68	94.32	95.91	97.07
Error rate in %	2.12	3.32	5.68	4.09	2.93

Message 150 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2081	2077	2076	2078	2083
Correct classificcaiton	2038	2008	1962	1995	2023
Missclassification	43	69	114	83	61
Success rate in %	97.93	96.68	94.52	96.01	97.07
Error rate in %	2.07	3.32	5.48	3.99	2.93

Message 300 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2081	2077	2076	2078	2083
Correct classificcaiton	2040	2009	1959	1996	2024
Missclassification	41	68	117	82	59
Success rate in %	98.03	96.68	94.37	96.06	97.16
Error rate in %	1.97	3.32	5.63	3.94	2.84

Message 600 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2081	2077	2076	2078	2083
Correct classificcaiton	2038	2009	1960	1995	2023
Missclassification	43	68	116	83	60
Success rate in %	97.93	96.68	94.42	96.01	97.12
Error rate in %	2.07	3.32	5.58	3.99	2.88

COVER images	All resolutions
Samples total	35000
Correct classificcaiton	34972
Missclassification	28
Success rate in %	99.92
Error rate in %	0.08

PQ algorithm, 13 hidden neurons, hyperbolic tangent in hidden neurons and saturated linear in output neuron.

Following (Figure 5) shows the training root mean square error (RMSE).

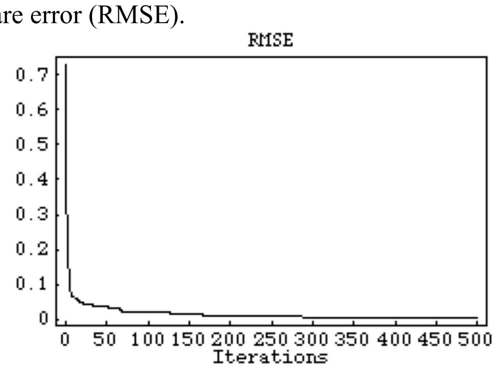

Figure 5: Example of training RMSE for 13 hidden neurons, hyperbolic tangent in hidden and saturated linear transfer function in output neuron.

Following tables (Tab. 10 – 18) deals with resolutions without two highest because of the space, the results are similar as in other resolutions.

Table 10 - 18: Results for PQ algorithm, 13 hidden neuron, hyperbolic tangent in hidden and saturated linear in output neuron, different length of messages and 5 resolutions.

Message 5 bytes	800 x 600	1024 x 768	1280 x 1024	1440 x 900	1680 x 1050
Samples total	2081	2077	2076	2078	2083
Correct classificcaiton	2047	2062	2050	2057	2068
Missclassification	34	15	26	21	15
Success rate in %	98.36	99.28	98.75	98.99	99.28
Error rate in %	1.64	0.72	1.25	1.01	0.72

Message 10 bytes	**800 x 600**	**1024 x 768**	**1280 x 1024**	**1440 x 900**	**1680 x 1050**
Samples total	2081	2077	2076	2078	2083
Correct classificcaiton	2047	2062	2050	2057	2068
Missclassification	34	15	26	21	15
Success rate in %	98.36	99.28	98.75	98.99	99.28
Error rate in %	1.64	0.72	1.25	1.01	0.72

Message 15 bytes	**800 x 600**	**1024 x 768**	**1280 x 1024**	**1440 x 900**	**1680 x 1050**
Samples total	2081	2077	2076	2078	2083
Correct classificcaiton	2045	2062	2050	2057	2068
Missclassification	36	15	26	21	15
Success rate in %	98.27	99.28	98.75	98.99	99.28
Error rate in %	1.73	0.72	1.25	1.01	0.72

Message 30 bytes	**800 x 600**	**1024 x 768**	**1280 x 1024**	**1440 x 900**	**1680 x 1050**
Samples total	2081	2077	2076	2078	2083
Correct classificcaiton	2045	2062	2050	2057	2068
Missclassification	36	15	26	21	15
Success rate in %	98.27	99.28	98.75	98.99	99.28
Error rate in %	1.73	0.72	1.25	1.01	0.72

Message 75 bytes	**800 x 600**	**1024 x 768**	**1280 x 1024**	**1440 x 900**	**1680 x 1050**
Samples total	2081	2077	2076	2078	2083
Correct classificcaiton	2046	2063	2050	2057	2068
Missclassification	35	14	26	21	15
Success rate in %	98.32	99.33	98.75	98.99	99.28
Error rate in %	1.68	0.67	1.25	1.01	0.72

Message 150 bytes	**800 x 600**	**1024 x 768**	**1280 x 1024**	**1440 x 900**	**1680 x 1050**
Samples total	2081	2077	2076	2078	2083
Correct classificcaiton	2045	2063	2050	2057	2068
Missclassification	36	14	26	21	15
Success rate in %	98.27	99.33	98.75	98.99	99.28
Error rate in %	1.73	0.67	1.25	1.01	0.72

Message 300 bytes	**800 x 600**	**1024 x 768**	**1280 x 1024**	**1440 x 900**	**1680 x 1050**
Samples total	2081	2077	2076	2078	2083
Correct classificcaiton	2046	2064	2050	2057	2068
Missclassification	35	13	26	21	15
Success rate in %	98.32	99.38	98.75	98.99	99.28
Error rate in %	1.68	0.62	1.25	1.01	0.72

Message 600 bytes	**800 x 600**	**1024 x 768**	**1280 x 1024**	**1440 x 900**	**1680 x 1050**
Samples total	2081	2077	2076	2078	2083
Correct classificcaiton	2047	2063	2050	2058	2068
Missclassification	34	14	26	20	15
Success rate in %	98.36	99.33	98.75	99.04	99.28
Error rate in %	1.64	0.67	1.25	0.96	0.72

COVER images	**All resolutions**
Samples total	35000
Correct classificcaiton	33644
Missclassification	1356
Success rate in %	96.06
Error rate in %	3.94

CONCLUSION

The paper deals with a steganalysis of PQ algorithm by means of artificial neural networks. The result section deals with 2 examples of simulation data from testing. The experiments were done for different settings of transfer functions – sigmoid, hyperbolic tangent, saturated linear in hidden and/or in output neuron. The number of hidden neurons was changed from 1 to 20. Testing simulations were carried out for 8 different resolutions and 9 length of inserted message. In some cases, even though RMSE was under 0.1, which should mean a good quality training, results during validation testing have high percentage of misclassification. One of examples can be the case of 15 hidden neurons with sigmoid transfer function and output function was set up to saturated linear. The overall misclassification was more than 10% which is not acceptable in the case of steganalysis. It is hard to say which settings is the best as in some cases are better classification for cover and in some for stego. Moreorless can be stated that sigmoid in hidden and sigmoid in output or hyperbolic tangent in hidden and saturated linear in output and 4 hidden neurons are a good solution for settings. Further research will continue with 2 layers neural nets if it will not help in this case with more length of inserted message. In the previous research with one type of hidden message 2 layer nets were not successful. This might be a more complicated study case and therefore more complicated nets will be a good solution.

AKNOWLEDGMENT

This work was supported by the grant NO. MSM 7088352101 of the Ministry of Education of the Czech Republic, by grant of Grant Agency of Czech Republic GACR 102/09/1680 and by the European Regional Development Fund under the Project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089.

REFERENCES

Cole E., Krutz D. R.: Hiding Sight, Wiley Publishing, Inc., USA, 321 s., 2003 ISBN 0-471-44449-9

Cormen, T.H., Leiserson, Ch. E., Rivest, R. L., Stein, C.: Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill, Section 16.3, pp. 385–392, 2001. ISBN 0-262-03293-7.

Freeman J. A.: Simulating Neural Networks with Mathematica, Addison-Wesley, Reading, MA, 1994

Neural Network Theory: Feedforward neural networks. Mathematica: Neural nets toolbox: Help

Fridrich, J., Goljan, M., and Hogea, D. "Steganalysis of JPEG Images: Breaking the F5 Algorithm." 5th Information Hiding Workshop, Noordwijkerhout, The Netherlands, Oct. 2002. URL: http://www.ws.binghamton.edu/fridrich/Research/f5.pdf. Last accessed: 2003-12-24.

Fridrich J., Goljan M., and Soukal D.: “Perturbed quantization steganography with wet paper codes,” in Proc. ACM Multimedia Workshop, Magdeburg, Germany, Sept. 20–21, 2004, pp. 4–15.

Goldwasser S., Bellare M.: Lecture Notes on Cryptography, Cambridge, 283 s., 2001

Hertz J., Kogh A. and Palmer R. G.: Introduction to the Theory of Neural Computation, Addison – Wesley 1991

Hetzl S.: Steghide (1) - Linux man page [online]. [cit. 2008-05-21]. available from WWW: <http://steghide.sourceforge.net/documentation/manpage.php>.

Matousek, R.: Using AI Methods to Find a Non- Linear Regression Model with a Coupling Condition. Engineering Mechanics, 2011, vol. 17, No. 5/ 6, p. 419-431. ISSN: 1802- 1484

Matousek, R. HC12: The Principle of CUDA Implementation. In MENDEL 2010. Mendel Journal series. 2010. Brno: VUT, 2010. p. 303-308. ISBN: 978-80-214-4120- 0. ISSN: 1803- 3814.

Oplatkova 2008a: Oplatkova, Z., Holoska, J., Zelinka, I., Senkerik, R.: Detection of Steganography Content Inserted by Steghide by means of Neural Networks, VUT v Brne, FME, MENDEL 2008 14th International Coference on Soft Computing, Brno, 2008, 166-171, ISBN 978-80-214-3675-6

Oplatkova 2008b: Oplatkova, Z., Holoska, J., Zelinka, I., Senkerik, R.: Steganography Detection by means of Neural Networks, IEEE Operations Center, Nineteenth International Workshop on Database and Expert Systems Applications, Piscataway, 2008, 571-576, ISBN 978-0-7695-3299-8

Oplatkova 2009: Oplatkova, Z., Holoska, J., Zelinka, I., Senkerik, R.: Detection of Steganography Inserted by OutGuess and Steghide by means of Neural Networks, AMS2009 Asia Modelling Symposium 2009, IEEE Computer Society, Piscataway, 2009, ISBN 978-0-7695-3648-4

Price, K. (1999), ‘An Introduction to Differential Evolution’, In: (D. Corne, M. Dorigo and F. Glover, eds.) New Ideas in Optimization, (pp. 79–108), London: McGraw-Hill

Zelinka I., “SOMA – Self Organizing Migrating Algorithm”, In: New Optimization Techniques in Engineering, (B.V. Babu, G. Onwubolu (eds)), chapter 7, 33, Springer-Verlag, 2004

AUTHOR BIOGRAPHIES

ZUZANA OPLATKOVA was born in Czech Republic, and went to the Tomas Bata University in Zlin, where she studied technical cybernetics and obtained her MSc. degree in 2003 and Ph.D. degree in 2008. She is a lecturer (Artificial Intelligence) at the same university. Her e-mail address is: oplatkova@fai.utb.cz

JIRI HOLOSKA was born in Czech Republic and went to the Tomas Bata University in Zlin, where he studied security technologies, systems and management. He obtained his MSc. degree in 2008. He is now studying a doctoral program on TBU in Zlin in the field of steganalysis and artificial inteligence. His e-mail address is: holoska@fai.utb.cz

IVAN ZELINKA was born in the Czech Republic, and went to the Technical University of Brno, where he studied Technical Cybernetics and obtained his degree in 1995. He obtained Ph.D. degree in Technical Cybernetics in 2001 at Tomas Bata University in Zlin. Now he is a professor (Artificial Intelligence, Theory of Information). Email address: ivan.zelinka@vsb.cz

ROMAN SENKERIK was born in the Czech Republic, and went to the Tomas Bata University in Zlin, where he studied Technical Cybernetics and obtained his MSc degree in 2004 and Ph.D. degree in Technical Cybernetics in 2008. He is now a lecturer at the same university (Applied Informatics, Cryptology, Artificial Intelligence, Mathematical Informatics). His email address is: senkerik@fai.utb.cz

ROMAN JASEK was born in the Czech Republic, and went to the Palacky University Olomouc, where he studied Informatics and obtained his degree in 1992. He obtained Ph.D. degree in Information Technologies at Charles University in Prague. Now he is an associate professor and head of department of Informatics and Artificial Intelligence at TBU in Zlin. His research interests: Information Security, Artificial Intelligence. Email address: jasek@fai.utb.cz