Protected streaming of video content to mobile devices

Miroslav Michalko, Tomáš Bobko, and Pavol Fogaš

Abstract— **Article describes the procedure of implementing designed architecture used for streaming of protected content as well as the process of application the protection itself on the streaming video content. Paper presents some of the basic components, as well as further advanced options of their use for the creation of the video streaming architecture basis. This is followed by the implementation of the architecture itself with the use of formerly chosen components and technologies where the explanation of the architecture components' functionality resides with the description of the video content protection method. Implementation consists of scenarios and applications for Apple iOS and Google Android platforms with the goal to present simple and low resources solution on server side and mobile device.**

***Index Terms*—streaming media, copyright protection, multimedia communication, Android, iOS, HLS**

I. Introduction

With a growing trend of capturing video and creating new applications for its spreading and playing on mobile devices, internet blogs and local network streaming at home, the importance of protection of this video against stealing arises as well. There are many ways to protect the video that it doesn't become a subject of stealing. Nowadays, simple but also sophisticated methods of how to protect a video exists, such as watermarking. By using this method, the overall quality of the video is worse but on the other hand, the size of the watermark itself will discourage the thief to steal the video, because of the length of the process to erase the watermark, so the video would be of a satisfied quality [1]. Another way to protect a video is to use steganography. A method that allows to embed a hidden message within chosen frames of the video. When the author would want to retrieve the message afterwards, he can easily do so. This solution is ineffective when applying video compression. The message from the video is changed or completely erased. That would mean that a thief could use only compression to get rid of the hidden message in the video and the video would still be of a good quality [12] [13]. Alternative of video protection described in this article resides in using a unique user identifier. The user is an authenticated subscriber, thus embedding this identifier within a video would mark the video of its subscriber. [11]

The goal of this work is to design an architecture that would let users stream videos protected against stealing with subscriber's identification. [2] The architecture should allow users to authenticate to a web server, store their sign in information within a database, then choose a clip to stream, embedding the subscriber's identification within the video and then stream it to the user. Database should also contain necessary information about video clips that are going to be streamed. [10]

With designed architecture, implement it and test the streaming of protected video content in browsers and an iOS mobile device or simulator.

The second goal of this work is to design and create system providing video on demand for client's Android mobile device. [19] System providing multimedia content must fulfill several requirements with regard to security. Provided content must be protected from unintended usage, such as unauthorized copying, and also this content must be protected from any kinds of attacks, with the intention to obtain content or its parts, during the transfer.

II. Methods for server side content protection

It is difficult to find a way of protecting multimedia content on the Internet. If the video can be played on a computer or another device, it means that the video can be stolen. The only steps that should be taken, is to try to make the video stealing unpleasant for the thief. [3] There are a few methods which worth consideration.

A. Digital Rights Management

A lot of digital content providers are selling their content not only on physical media, but also through computer networks. Without content protection and digital rights

Manuscript received August 10, 2014. This paper is the result of the Project implementation: University Science Park TECHNICOM for Innovation Applications Supported by Knowledge Technology, ITMS: 26220220182, supported by the Research & Development Operational Programme funded by the ERDF.

M. Michalko. Author is with the Computer Networks Laboratory at Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Kosice, Letna 9, Kosice, Slovakia, phone: +421(55)6027080; e-mail: miroslav.michalko@tuke.sk.

T. Bobko was with Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Kosice, Letna 9, Kosice, Slovakia, phone: +421(55)6027075; e-mail: kalafun@gmail.com.

T. Bobko was with Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Kosice, Letna 9, Kosice, Slovakia, phone: +421(55)6027071; e-mail: avol.fogas@student.tuke.sk.

management, can be this content easily copied, edited and spread further to a wide audience [14] which would make a hole in the profit. The DRM systems can be used to protect valuable properties and management of their distribution and usage. A right DRM system should provide protection against unauthorized access to the digital content, limiting the access only for those with proper authorization. The main DRM component is the use of digital licenses. Instead of buying the digital content, the subscriber only buys the license that will grant him certain rights. A license is a digital file, which determines certain rules how to use the digital content. [5]

B. *Watermarking*

A digital watermark is a signal that can be embed into a digital content for various reasons for example subtitling, copyright control. An important property of a watermark is their robustness in common signal changes as the file filtering and file compression. A next use of a watermark is embedding notes into files and access control. These watermarks are called "annotation watermarks". As an example, the rule of using content, that defines allowed copies and replays, can be embedded as an annotation watermark into every single copy of the content. The usage check is implemented within the user's player.

C. *HTTP Live Streaming*

For streaming and playing videos on and iOS device, the HLS is used. It is a communication protocol for streaming media based on HTTP, which was implemented by Apple [6]. HLS is sending audio and video a way that it splits the content into multiple ten seconds parts also called media segment files. Index file or a playlist gives users the URL of these segments. The playlist can be regularly renewed so it contains the URLs of the current media segment files. [21]

D. *Steganography*

The word "steganography" is of Greek origin, the words "hidden writing" and means "hide for common sight". [4] Steganography is science and an art communicate the way that presence of the message can't be revealed. Some simple techniques are used hundreds of year, but with growing use of digital files, there are some new techniques for information hiding. [7]

E. *Streaming solution using Wowza Streaming Engine*

Wowza Streaming Engine provides a new generation software for streaming media that simplifies high quality live streaming as well as streaming video on demand for all devices connected to the internet. There are multiple advantages why to use the Wowza Streaming Engine. Streaming for multiple devices with high quality live streaming and video on demand streaming. The configurability for various operating systems and architectures, from one computer or a load-balanced server or cloud. Extendable API in Java makes writing own modules and extensions easier. [8]

III. Secure system for protected delivery over network

The label "secure system" involves numerous requirements. We can say that the main components, in terms of ensuring the transmission of content and content itself are conditional access, authentication, copy protection, content watermarking and secure transmission of content. [1]. One of the methods ensuring security of any content transmission is encryption. It is a process of transforming any information into form, from which those information cannot be read or understand the meaning of the information. The opposite process is called decryption.

In this process, the person for whom are those data intended, can obtain original information. For encryption and decryption is usually used some key or password. Scheme showing principle of usage encryption in communication between two parts is shown in Fig. 1. There are multiple encryption algorithms and we decided to use AES (Advanced Encryption Standard) in our system. (AES is a block encryption algorithm. It supports variable key lengths of 128, 192 and 256 bits. [2]) [27] AES is currently one the most commonly used encryption algorithms in video streaming protocols, such as HLS (HTTP live streaming), which we will use in our system. HLS is a streaming protocol allowing providing video content, both live and video on demand. [16] HLS consists of three main parts: server part, distribution part and client software. Server part handles video content, and prepares it for distribution. Distribution part handles requests from users and delivers requested video content. [6] Client part receives this video content, processes it and displays to the user. Gábor Fehér in his work [3] showed, that the mobile devices, used in his tests, were able to smoothly handle and play video stream encrypted with AES. In the results of measurements we could see that the increase of CPU load, processing encrypted stream compared to unencrypted stream only increased by 7.39 percent. This measurement also showed that the video stream was more limited by the network bandwidth than computational processes carried out in a mobile device. [22]

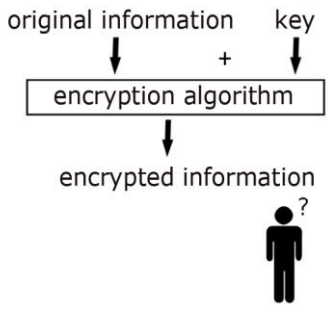

Fig. 1 Encrypting content

IV. Architecture for server side content marking

To implement content protection of streaming media, it is necessary to design an architecture that will provide streaming digital content which will carry the protection mark, thus the unique identification of a subscriber. This architecture will consist of these components:

Webserver - user interface to register, sign in, choose a video clip to stream, and finally watch the stream within a browser,

Database - a storage for all the user data including their email address, hashed password, registration timestamp and also the video clip data storage. Every clip should have its name, a stream that he belongs to and some other optional data too, like timestamp of last streaming or creation,

Digital content storage - physical storage to store the video clips in. This storage will be required by the streaming server,

Streaming server - this is where the Wowza Media Engine will reside with all its features. This server will receive and handle all requests from users to stream protected content according to user,

Mobile device - user's mobile device in this case with iOS operating system. The architecture is defined in Fig. 2.

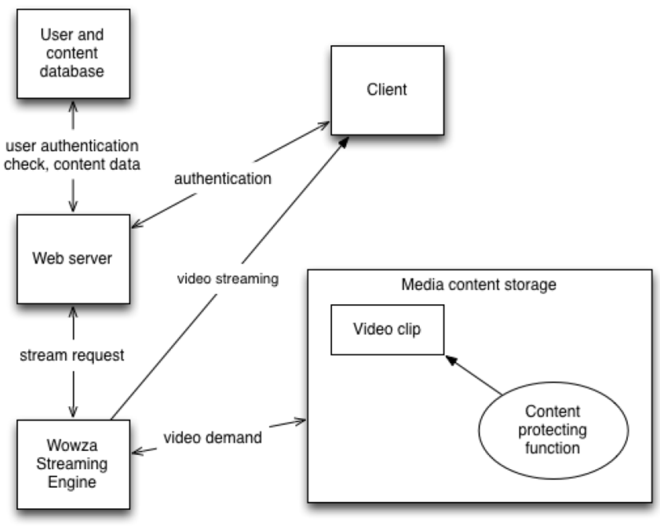

Fig. 2 Streaming architecture

Client sends a request from a web browser to register himself. Webserver communicates with the user database and the database will store user's email and password. If it’s a mobile device, the registration will not be available, user can sign up only through a browser. If the user will be already registered and would want to sign in, he should do so using his iOS application or a browser. The process will be similar to registration, but instead of creating a row in the database, the web server will only compare the data from user and database. [15]
When the registration is successful, user will be redirected to the page where he can choose which clip he wants to stream. Data for the choose view will be fetched from the clips table from database. After choosing which clip should be played, the Wowza Streaming Engine will apply the function for protecting the content and starts to stream the protected content to the subscriber. The content protecting function is defined in the Fig. 3.

The first thing to start can be the installation of the Wowza Streaming Engine. An exact procedure how this should be done can be found on the Wowza Streaming Engine User Guide. After successfully installing Wowza, the Wowza manager can be run via browser to set up new streams. The Wowza scheduler needs to be implemented as well so the stream could be segmented into pieces with the video content and the subscriber's identification. Also the HTTPProvider module should be implemented, that will respond to http requests with a stream URL. When this is set, the implementation of the user's database with two tables’ users and clips, where appropriate columns will be set like email, password, user hash for users table and clip name, stream name for clips. Web server can be implemented in any known language that can work with JSON [28].
The process of streaming will follow these simple steps:
1. Registration or authentication of the user,
2. After choosing a clip to stream, a request is sent to Wowza Streaming Server with user hash, thus user's unique identification
3. Wowza's HTTPProvider module fetches this request and runs a script that will generate a video using the mpeg tool with user hash in it and also a SMIL file [9], so the scheduler will know how to mix the video clip with the video of the user hash, the HTTPProviders returns a stream URL for the client's device,
4. When the client fetches the respond, the stream player can be initialized with the given URL and start to stream the protected content.

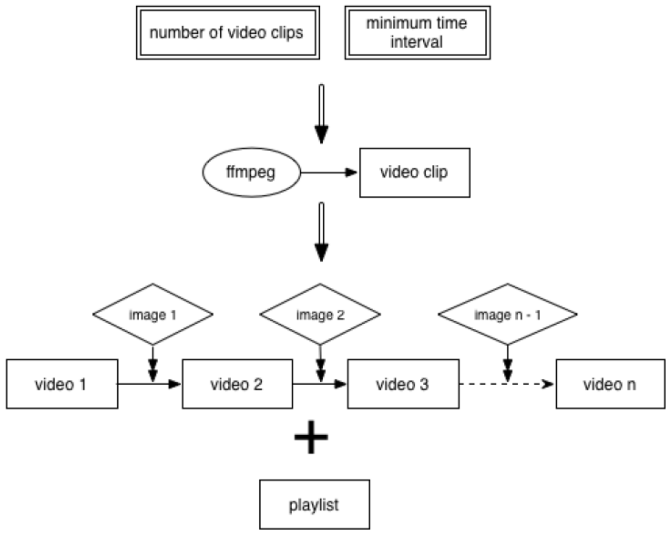

Fig. 3 Content protecting function

V. Solution for delivery to Android powered devices

A. *The design of system providing multimedia content in secure way*

Results of the analysis lead us to design and implementation of system providing content in secure way. Our system consists of three main parts: Web application, Wowza part and Android application. The purpose of web application is to accomplish the following tasks:

- user and device management,
- multimedia content management,
- user requests processing and evaluation,
- user authentication,
- providing links to multimedia content.

The next component is Wowza server. Main purpose of this component is to transfer multimedia content to client devices. This content will be transferred via HLS. This protocol was chosen for several reasons, mainly because the data is transmitted via HTTP, also it is possible to use encryption algorithm AES to encrypt video content. Streaming protocol HLS is officially supported by Android platform, but in reality, it is not working well. [17] [18] [20] HLS is mainly intended for Apple devices, as it is developed by Apple, so another challenge is to create reliable Android application that handles encrypted HLS streaming. Multimedia framework Vitamio was chosen for displaying video content to user. In the following Fig. 5 is shown design of this scheme.

Using Petri nets it is possible to make formal defining of structure and behavior of various systems. [24][25][26] Petri net is a triplet

$N = (P, T, A),$ where

$P = \{p_1, p_2, ..., p_n\}$ is a finite non-empty set of places,

$T = \{t_1, t_2, ..., t_n\}$ is a finite set of transitions,

A is the set of oriented edges, it being understood that the edges may be associated only places with transitions, which can be formally written as

$$F \subseteq (S \times T) \cup (T \times S).$$

Petri net can be represented as a bipartite graph, which are shown as circles of interest, the transitions are represented as boxes and arrows as directed edges. In this case it is used for system design described above. Petri net is shown on Fig. 4.

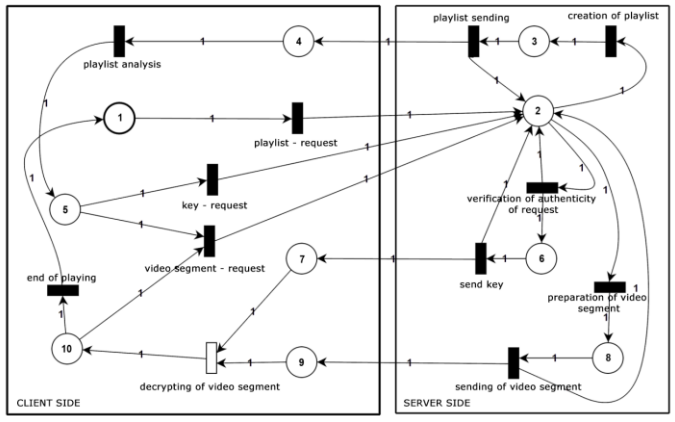

Fig. 4 Petri net representing delivery to Android based player

Numbers represents:

1. Initial state, the user chose video.
2. Requirements adopted and evaluated.
3. The list is created for playback. Also referred to as a playlist.
4. The list of playback is adopted.
5. List for playback is processed.
6. Encryption-decryption key was created.
7. Encryption-decryption key was adopted.
8. Video segment was created.
9. Video segment was adopted.
10. Playing a video.

In our case, therefore, can be expressed as a set of places

P={1,2,3,4,5,6,7,8,9,10}

and since the transitions are labeled with the words, expressed as a set of transitions

T = {playlist -request, creation of playlist, playlist sending, playlist analysis,

key – request, verification of the authenticity, send key,

video segment – request, preparation of video segment, send video segment, decrypting of video segment, end of playing}.

The Petri net consists of two units, representing the client side and the server side. Diagram shows the process that takes place between how a user selects a video to be played and the starting player. Left side shows the client side and the right side shows the server side.

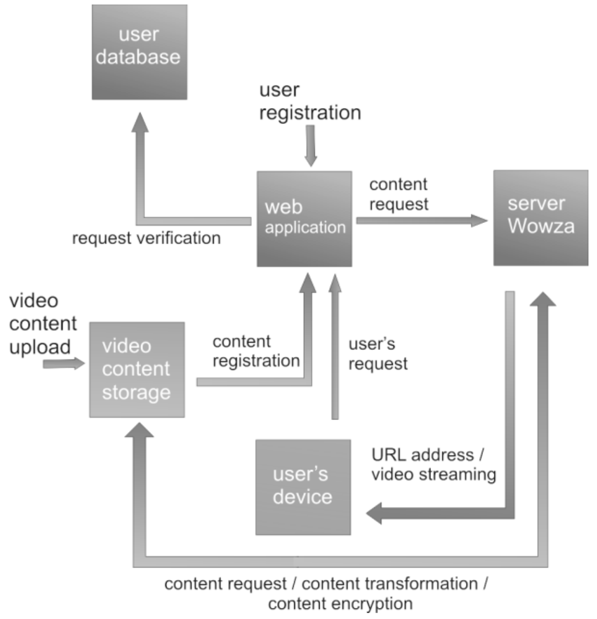

Fig. 5 System providing multimedia content in secure way.

B. *The design of secure content transfer*

After the client device, smartphone or tablet gets address of multimedia content, it sends a request to the Wowza server. This server returns playlist in .m3u8 format, which contains information about required multimedia content segments. Segments are downloaded one after one, and also the key, that is needed to decrypt them, since these segments are in encrypted form. Diagram in the following Fig. 6 illustrates this process.

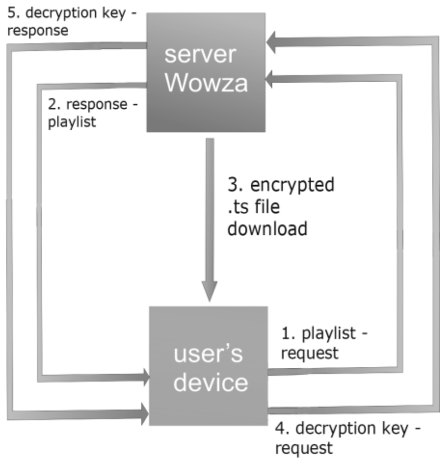
5. decryption key - response
server Wowza
2. response - playlist
3. encrypted .ts file download
1. playlist - request
user's device
4. decryption key - request

Fig. 6 The design of secure content transfer.

C. *The design of Android application*

The design involves the creation of a layer between the player and the encrypted content delivery. This layer will receive the list of encrypted .ts files and also corresponding decryption key, which is downloaded from server in secure way using HTTPS protocol. These encrypted video segments are transferred to the player that will encrypt them and display to the user. In the following Fig. 7 is shown design of this scheme.

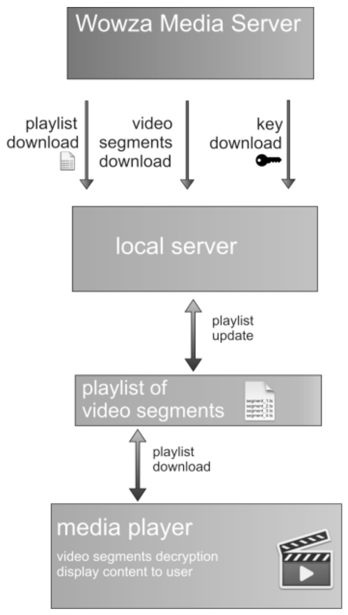

Fig. 7 Android application architecture.

VI. Results

As result one server side FFmpeg based encoder and two mobile applications were developed. End mobile users were asked to test the solution in real conditions on their mobile phones and to fill in simple questionnaire to map their subjective remarks. The implemented application, which home screen is shown in Fig. 5, was tested on seven different devices. List of tested devices is in Table 1. In the Table 2 are shown results of testing this application in the form of user's reviews.

TABLE 1
TESTED DEVICES

Device	Processor	RAM	Screen resolution
Samsung Galaxy S4	quad core 1,9 GHz	2 GB	1080 x 1920 px
Samsung Galaxy S4 mini	dual core 1,7 GHz	1,5 GB	540 x 960 px
Motorola Moto G	quad core 1,2 GHz	1 GB	720 x 1 280 px
Motorola Fire	600 MHz	256 MB	240 x 320 px
ZTE Blade III	1 GHz	512 MB	480 x 800 px
Sony Xperia L	dual core 1 GHz	1 GB	480 x 854 px
HTC Desire Z	800 MHz	512 MB	480 x 800 px

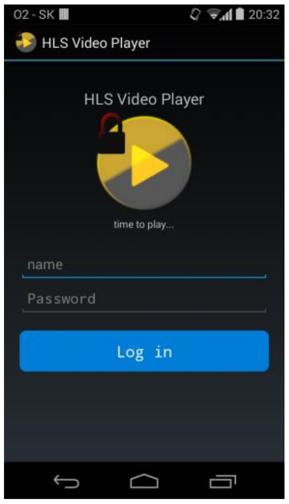

Fig. 8 HLS Player - Android application UI.

TABLE 2
USERS REVIEWS

Device	Application Design	SD video playback	HD video playback
Samsung Galaxy S4	satisfied	without problem	Audio/Video not synchronized
Samsung Galaxy S4 mini	satisfied	viewable	viewable
Motorola Moto G	satisfied	without problem	viewable
Motorola Fire	unsatisfied	viewable	unviewable
ZTE Blade III	satisfied	without problem	not tested
Sony Xperia L	satisfied	without problem	viewable
HTC Desire Z	unsatisfied	without problem	unviewable

As Android devices are so fragmented and various in hardware specifications it is necessary to make further development in the created mobile application. Some automatic detection of supported screen resolutions and CPU speed could be added. Overall satisfaction of users involved in testing was good based on feedback provided.

Testing was done on 27 users with various mobile Android OS powered devices, 55% of them were satisfied with functionality provided by application, 30% lacks important functionality but were positive or neutral to application and 15% were disappointed. Feature where application stores position of paused video playback and synchronizes it between other registered devices was mostly commended by all participants involved in testing. None of user have seen some watermarking or hidden identification stored to video stream what was the goal of this work. Further testing will be focused to comparison of resources needed on server side in compare to similar market solutions.

VII. Conclusion

The goal of this work was to design an architecture that would stream protected content to its mobile clients, using some of the well-known technologies and standards.

Several conclusions could be made based on presented work:

- insertion of visible marking to video (one frame to 25fps video) is not visible to normal viewer but provides satisfactory higher level of source identification and protection due to time when real time steganography will take place in real-time streaming applications;
- problem oriented applications for mobile smartphones provides practical platform as extra extension to secure content delivery of multimedia;
- no extra processing resources or new streaming technologies are needed to insert content providers identification into stream due to utilization of user oriented playlists;
- SMIL standard could be revised in order to provide language profiles for secure content identification for streaming servers;
- HLS should be implemented to wider number of platforms as standard for secure streaming.

The design presented in this paper describes how a streaming system could be built however, some other standards could be used instead of SMIL or a different streaming server instead of Wowza Streaming Engine. The advantages of using this method are that most of the technologies used are for free and can be interconnected. The FFmpeg tool [23] could be replaced with another tool of similar functionality, to create a short video with an embedded text. There could be some automation in the HTTPProvider section, because of the need of restarting the streaming server after each request for streaming due to the Wowza Streaming Engine scheduler module. This disadvantages will be improved by the next development. In content delivery over IP network standard procedures and techniques such as SSL, AES, etc. were used. The challenge was to create a mobile app for Android platform that would play encrypted content delivered via HLS. The application has been tested by a group of users, using different devices. Paper presents base for further development and testing.

Acknowledgment

Paper is the result of the Project implementation: University Science Park TECHNICOM for Innovation Applications Supported by Knowledge Technology, ITMS: 26220220182, supported by the Research & Development Operational Programme funded by the ERDF.

References

[1] M. Staněk. (2004, January 6). Cryptology Basics [Online]. Available: https://fmfi-uk.hq.sk/Informatika/Kryptologia/prednasky/krypto.pdf

[2] Eugene T. Lin. *An Overview of Security Issues in Streaming Video*. In Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC '01). IEEE Computer Society, Washington, DC, USA, 2001, pp. 345

[3] Feher, G., "*The Price of Secure Mobile Video Streaming*," Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on , vol., no., pp.126,131, 25-28 March 2013

[4] J. Cummins,et al. "*Steganography and digital watermarking*." School of Computer Science, The University of Birmingham 14, 2004.

[5] Q. Liu, S.N. Reihaneh, N.P. Sheppard. "*Digital rights management for content distribution*." Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003-Volume 21. Australian Computer Society, Inc., 2003. pp. 49-58.

[6] Apple Inc. (2014, February 11). HTTP Live Streaming Overview [Online]. Available: https://developer.apple.com/library/ios/documentation/networkinginternet/conceptual/streamingmediaguide/StreamingMediaGuide.pdf

[7] S. Das, et al. "Steganography and Steganalysis: different approaches." arXiv preprint arXiv:1111.3758, 2011.

[8] Wowza Media Systems. (2010, February 12). Wowza Streaming Engine – Users Guide version 4. [Online]. Available: http://www.wowza.com/forums/content.php?3-quick-start-guide

[9] P. Hoschka. (2011, April 19). SMIL – An Introduction [Online]. Available: http://www.w3.org/2002/05/siggraph-smil-abstract.pdf

[10] L. Yao, L. Fei, L. Guo, S. Bo, Ch. Songqing, "*A server's perspective of Internet streaming delivery to mobile devices*," INFOCOM, 2012 Proceedings IEEE , vol., no., 1332,1340, March 2012, pp. 25-30

[11] D.M. Chen, S.S. Tsai, R. Vedantham, R. Grzeszczuk, B. Girod, "*Streaming mobile augmented reality on mobile phones*," Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on , vol., no. 181,182, Oct. 2009, pp. 19-22

[12] L. Vokorokos, A. Pekar, P. Fecil'ak, "*IPFIX Mediation framework of the SLAmeter tool*," Emerging eLearning Technologies and Applications (ICETA), 2013 IEEE 11th International Conference on , vol., no., 311,314, Oct. 2013, pp. 24-25

[13] M. Kriška, J. Janitor, P. Feciľak. "*Dynamic routing of IP traffic based on QoS parameters*", International Journal of Computer Networks & Communications (IJCNC), Vol. 6, No. 4, July 2014, pp. 11-22

[14] R. Horak, J. Hrbacek, "*Elearning and mobile devices - Technical problems and possible solutions*," Emerging eLearning Technologies and Applications (ICETA), 2013 IEEE 11th International Conference on , vol., no., 123,126, Oct. 2013, pp. 24-25

[15] D. Cymbalak, F. Jakab, M. Michalko, "*Next generation IPTV solution for educational purposes*," Emerging eLearning Technologies and Applications (ICETA), 2011 9th International Conference on , vol., no., 41,46, Oct. 2011, pp. 27-28

[16] R.M. Schmitt, T.R. Muck, AA. Frohlich, "*An Implementation of the AES Cipher Using HLS*," Computing Systems Engineering (SBESC), 2013 III Brazilian Symposium on , vol., no., 113,118, Dec. 2013, pp. 4-8

[17] Z. Xueliang, T. Dan, "*The architecture design of streaming media applications for Android OS*," Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on , vol., no., 280,283, 22-24 June 2012, pp. 22-24

[18] T. Mantoro, M.A. Ayu, D. Jatikusumo, "*Live video streaming for mobile devices: An application on android platform*," Uncertainty Reasoning and Knowledge Engineering (URKE), 2012 2nd International Conference on , vol., no., pp.119,122, 14-15 Aug. 2012

[19] D.T. Massandy, IR. Munir, "*Secured video streaming development on smartphones with Android platform*," Telecommunication Systems, Services, and Applications (TSSA), 2012 7th International Conference on , vol., no., pp.339,344, 30-31 Oct. 2012

[20] D. Mrozek, B. Buk, B. Malysiak-Mrozek, "*Some remarks on choosing video stream encoding for remote video verification on Android mobile devices*," AFRICON, 2013 , vol., no., 1,6, Sept. 2013, pp. 9-12

[21] K. Lazic, M. Milosevic, G. Miljkovic, N. Ikonic, J. Kovacevic, "*One Implementation of Dynamic Adaptive Streaming over HTTP*," Telecommunications Forum (TELFOR), 2012 20th , vol., no., pp.1496,1499, 20-22 Nov. 2012

[22] S. Verma, S.K. Pal, S.K. Muttoo, "*A new tool for lightweight encryption on android*," Advance Computing Conference (IACC), 2014 IEEE International , vol., no., pp.306,311, 21-22 Feb. 2014

[23] F. Bellard. (2004, December 19). FFmpeg project [Online]. Available: https://www.ffmpeg.org

[24] W. Reisig, G. Rozenberg, eds. *Lectures on Petri nets I: basic models: advances in Petri nets*. Vol. 1491. Springer, 1998.

[25] K. Jensen, L.M. Kristensen, L. Wells. "*Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems.*" International Journal on Software Tools for Technology Transfer 9.3-4 (2007): 213-254.

[26] G. Rozenberg, P. S. Thiagarajan. "*Petri nets: Basic notions, structure, behaviour*." Current trends in concurrency. Springer Berlin Heidelberg, 1986. 585-668.

[27] S. Frankel, R. Glenn, S. Kelly. *The AES-CBC cipher algorithm and its use with IPsec*. RFC 3602, September, 2003.

[28] D. Crockford. "*The application/json media type for javascript object notation (json).*" 2006.

Miroslav Michalko is an Assistant Professor at Technical University of Košice. He obtained a PhD. (2010) in Informatics and the Masters Degree in Computer Science at Technical University in Kosice. Now he lectures Computer Networks and Application of Computer Networks. His research areas interests are in multimedia technologies, streaming and content delivery over IP, multimedia content creation and innovative teaching&learning techniques.

Tomáš Bobko received Masters Degree at Technical University in Košice (2014) in Computer Science. Now he works as software developer and consultant for private SME. This article partially presents results of his diploma work.

Pavol Fogaš received Masters Degree at Technical University in Košice (2014) in Computer Science. Now he works as software developer and consultant for private SME. This article partially presents results of his diploma work.