Information & Security

I. Iliev & I. Blagoev
vol. 53, no. 1 (2022): 78-88
https://doi.org/10.11610/isij.5306

Published since 1998

ISSN 0861-5160 (print), ISSN 1314-2119 (online)

Research Article

An Approach to Improve Web Video Streaming Security and Prevent Personal Data Leakage

Iliyan Iliev (✉), *Ivan Blagoev*

Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Sofia, Bulgaria, http://www.iict.bas.bg/EN

ABSTRACT:

The invention of television changed the world and made information spread faster. For the past few years, broadcast television has remained digital only and IP networks have emerged as an additional broadcast option. Regardless of the infrastructure used by a given broadcaster, some channels or even special content require a specific payment scheme and implement conditional access. This study proposes an approach for secure video streaming and preventing video content from easily leaking due to cyber security issues. The proposed methods cover both aspects – the content itself and the protection of subscribers' personal data when using live streaming services.

ARTICLE INFO:

RECEIVED: 17 JULY 2022
REVISED: 05 AUG 2022
ONLINE: 21 SEP 2022

KEYWORDS:

cybersecurity, live streaming, IPTV, RTMP, HLS, authentication

Introduction

The invention of television changed the world and made the spread of information much faster. Usually, a given television production defines a theme, such as sports, news, music, movies, etc., which is broadcast on a respective television channel.[1] The set of channels forms a package that is offered and sold to the end user by the broadcaster and distributed via terrestrial, cable, or satellite communications in traditional broadcast services. In the last few years, broadcast TV has remained digital only and IP networks have been gaining ground as an additional broadcast option.

✉ Corresponding Author: E-mail: iliyan.iliev@iict.bas.bg

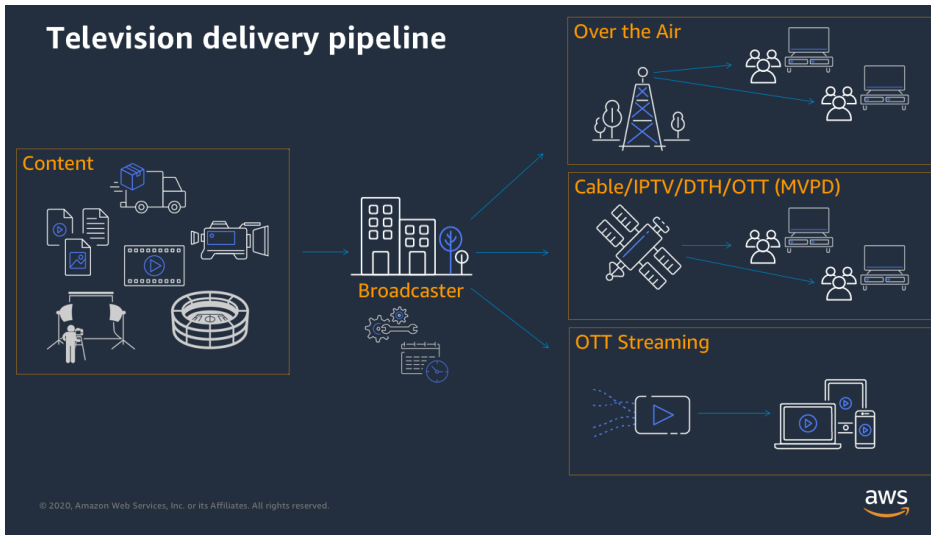

Figure 1: Television delivery pipeline (source Amazon).

Regardless of the infrastructure used by a given broadcaster, some channels or even special content require a specific payment scheme and apply conditional access. This means that only subscribers to a service can watch the paid channels and no one else. In the past, during the analog television era, broadcasters used coding techniques such as Video cipher to protect content (Fig.2). The customer must plug in a decoder to return the audio/video (AV) signals to their original form.

In the era of digital television, coding schemes evolved into encryption schemes such as Viaccess, Irdeto, Conax and many others. Customers must possess a smart card that keeps the user's identity secure and works in decryption devices, such as a conditional access module (SAM) on a TV or set-top box (STB), conforming to the encryption scheme.

The growth of the Internet allows many professionals and amateurs to expand their digital business on a global scale. Many communities and corporations develop specific communication protocols, security mechanisms, hardware devices, and software platforms for different types of activities and implementations. Increasing Internet speeds make video streaming and real-time viewing possible from many locations. As a result, conventional TV channels are facing new competition in the field of web video services. So some organizers of sporting events, concerts, conferences, etc. prefer web streaming production over traditional television coverage for an event. This trend is mostly observed in low-budget events. Although there is no direct technical relationship between budget and quality of service, in fact a number of problems and vulnerabilities can be observed simultaneously caused by low budget. The present research is not aimed to deal with the creation of the video content but only with security considerations regarding the upstream and

Figure 2: Video Cipher scrambler device analog content protection for TV operators.

downstream streaming process, whose improvements would pave the way for the development of these services.

Real-Time Messaging Protocol (RTMP)

Real-Time Messaging Protocol (RTMP) is the most common protocol for streaming video content in real-time. This protocol was developed by Adobe in an initial version for the transmission of multimedia to consumer Flash players and later further developed as the main protocol for live broadcasts on the Internet. RTMP live streaming is supported as functionality in a range of software and hardware video encoders for hobbyist to professional use. On the server side, RTMP is supported as a module in some of the famous servers (e.g., nginx) and in the popular public video sharing platforms – Face Book, Instragram, Youtube, TikTok, Twitch, etc.[1,2]

The sources of audio and video signals, which can be cameras, microphones, musical instruments, mixing consoles, pre-recorded multimedia, etc., are processed into a single video stream, after which it is encoded according to an appropriate transfer scheme (eg H. 264) with the necessary parameters for AV compression. The output stream is sent over RTMP to an RTMP endpoint, which can be hosted on a dedicated server or a public video-sharing platform. Setting up your own RTMP server allows implementing high traffic protection as well as uncovering a number of opportunities to deploy the further infrastructure until the content reaches the end client, such as e.g. traffic cloning to other RTMP endpoints for cross-platform simulcasting or for CDN provisioning, i.e. achieving geographically closer service to the end user.

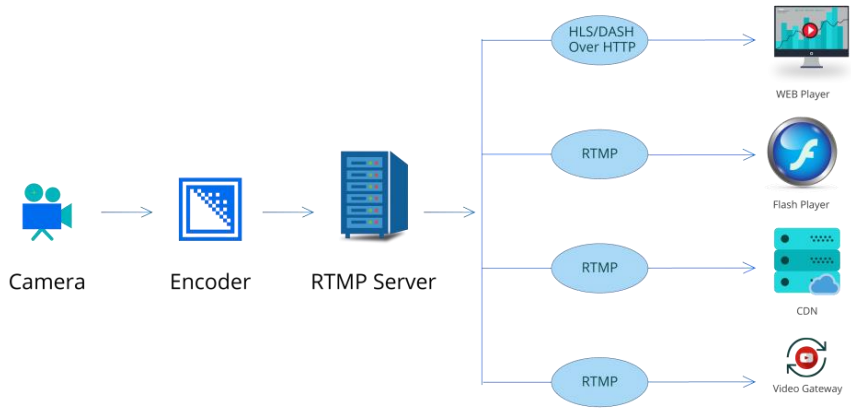

Figure 3: RTMP streaming processing diagram (source synopi.com).

In terms of cyber security and low-layer transport protocols, RTMP exists in 5 variants:

- RTMP - native protocol form of standard TCP port 1935. Can be omitted in RTMP URL;
- RTMPS – RTMP over TLS. There is no standard TCP port defined, 1936 is often used and must be explicitly mentioned in the RTMP URL;
- RTMPE – RTMP with Adobe encryption mechanisms;
- RTMPT – RTMP over HTTP(S) (when the provider has blocked all ports to destinations except 80 and 443);
- RTMFP – P2P UDP communication

In practice, the first two variants are used, namely RTMP and RTMPS. The first aspect of the article, looking at the protection of specific video content when it is only intended for a strictly defined group of subscribers, is related to ensuring the security of the communication between the streaming studio and the RTMP endpoint when the traffic passes through a public network like the Internet. In this case, using RTMPS is not always a good idea.[1] The reason is that software programs, or streaming hardware, generally do not have the ability to make specific TLS settings, as well as a reliable source of random numbers.[3] The best way to protect the traffic is to pass through a pre-established cryptographic tunnel, e.g., IPSEC between the streaming studio and the RTMP endpoint. Thus, RTMP traffic in its pure form can be carried through the tunnel without problems.[5,6] RTMPS should only be used when a cryptographic tunnel cannot be established in advance for one reason or another, most often financial.

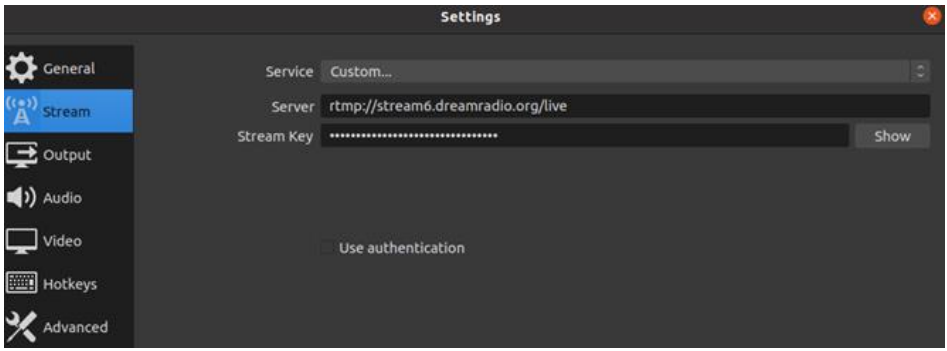

Figure 4: HTTP Live Streaming Protocol (HLS).

HTTP Live Streaming Protocol (HLS)

Although RTMP also allows viewing of the stream from the same endpoint it is being streamed to, this is never used in practice for end users. The reasons are:

- A limited number of players support RTMP communication;
- RTMP requires a constant TCP connection, which would cause problems for live viewing through over 90% of conventional home ISP's and mobile operators;
- RTMP does not have flexible end-user authentication mechanisms.

The three problems have been solved with the introduction of a new HLS protocol developed by Apple, which is only intended for viewing video content by end users:

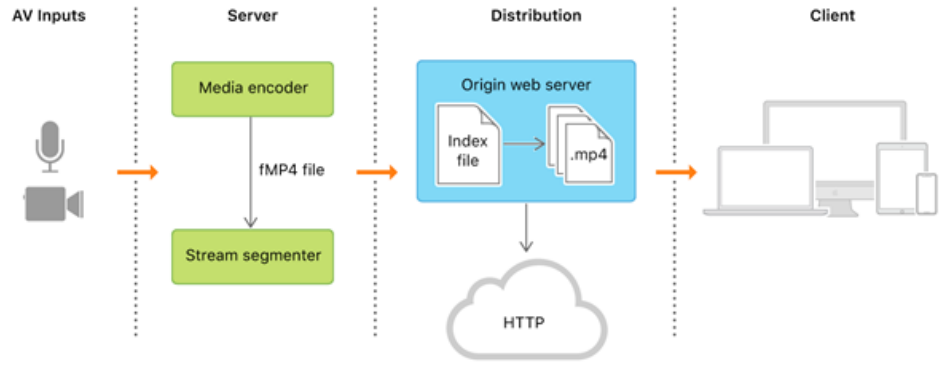

Figure 5: HLS Architecture (source: Apple).

Where the stated issues are resolved:

- It is supported by almost all HTML5 players and browsers;

- The real-time stream (RTMP in this article) is converted into HLS chunks, which are served over HTTP(S) to the end user;
- All HTTPS connection security and authentication mechanisms can be applied.

To look at the traffic protection and user authentication mechanisms, it's need to look deeper into HLS. This can happen after starting a working HLS viewer endpoint, which can be implemented by performing the following steps:

- Setting up an RTMP server to receive RTMP traffic on the appropriate endpoint;
- RTMP to HLS conversion setup.

Example with nginx configuration snippet with compiled nginx-rtmp-module for steps 1. and 2: https://github.com/arut/nginx-rtmp-module then, follows:

- Encode incoming AV signal to H.264 (using streaming program or specific hardware device;
- Entering the RTMP endpoint and the corresponding Stream Key in the program or device settings;
- Start the streaming process;
- Development (clone from the repository) of a web player enabling HLS viewing.

Sample code of the above: hls.js project, https://github.com/video-dev/hls.js/

- Entering the HLS settings in the corresponding JS file;
- Embed the player in a basic web page;
- Play the video from the player and open the Developer console of the browser to view the traffic.

The player's networking is reduced to making asynchronous XHR requests to the HLS endpoint, which are divided into 2 types. The first type of request downloads an m3u8 playlist with Content-Type: application/vnd.apple.mpegurl, containing single points of the available chunks, and the second type downloads the corresponding chunk with Content-Type: video/mp2t.[4]

The screenshots show portions of sample response content for both types of requests (Figure 7).

Here comes the moment of the second aspect of the research, which concerns the video distributor and looks at the protection of the video content towards the end users. Since specific video content is intended for a strictly defined group of users, it follows that the HTTPS server must first be configured according to security best practices (https://www.ssl.com/guide/ssl-best-practices/). After that, user authentication mechanisms must be implemented. The options generally boil down to two main ones:

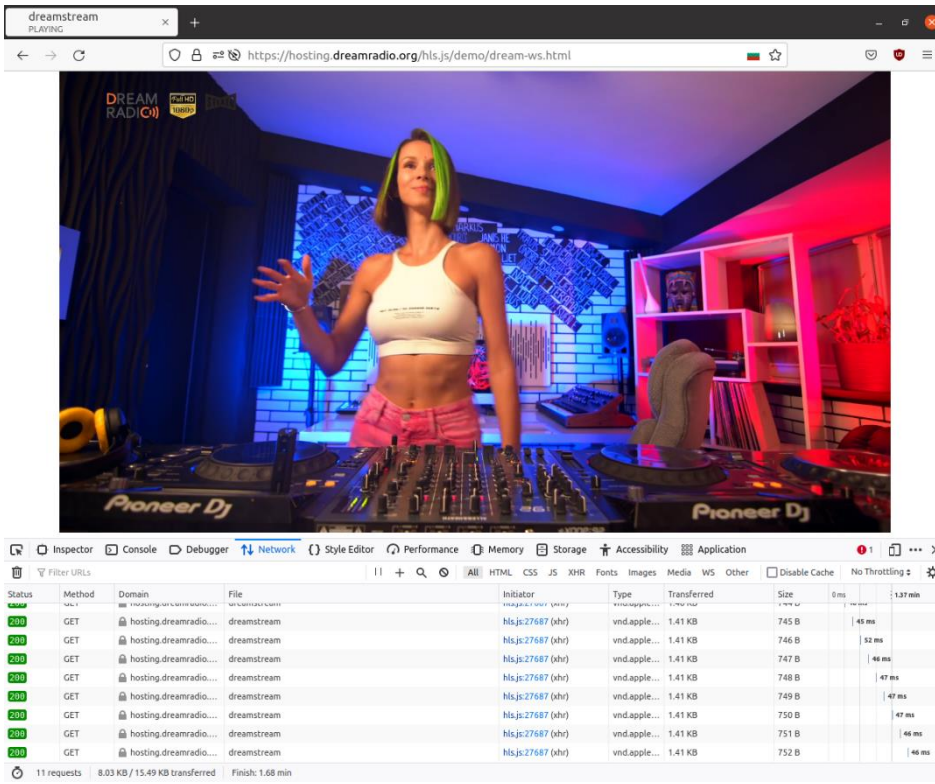

Figure 6: Video streaming code page source decoding.

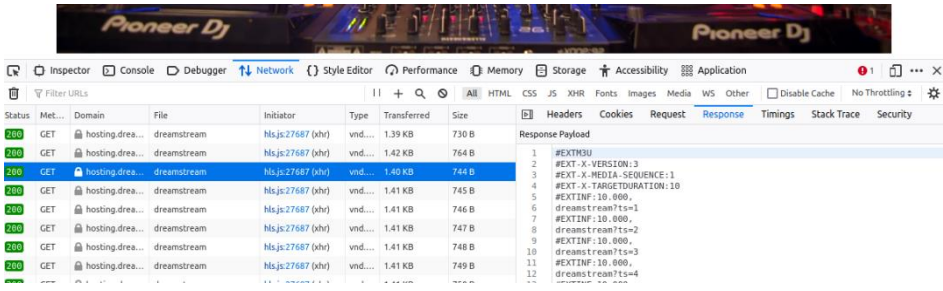

Figure 7: Video streaming code page source decoding.

- Establishing TLS with a requirement for client authentication with a certificate;
- Managing application user sessions with cookies or a similar method.

The first option is used extremely rarely because it is more complicated from the user's point of view. A valid user TLS authentication certificate signed by in

a certificate chain is required.[6] As often, it also requires a specialized crypto-processor to protect the user's private key to be implemented properly. Although quite secure, the complexity of the many elements in it, which are mostly transferred to the user, makes it not widely used for this purpose.

With the second option, the chunks, as well as the playlist, cannot be downloaded by any user on the Internet who has their URL addresses, but only by authorized ones who can transmit a valid token to download the content, in other words – those who have passed authentication and authorization processes. In the world of video distribution, an authorized user is most often understood as one who has access to a given content after payment has been made.

For the needs of the study, a simulator of such processes was developed in the backend, functioning between the "bare" HLS service of nginx and the end user, which takes care of providing a token to the user in a cookie after valid authentication and, accordingly, validates the tokens from the client requests. If the token is not submitted or is invalid (expired), the backend returns 401 Unauthorized.[7]

The possibilities of obtaining a token are numerous and varied. They mostly depend on the management decisions for the organization or group of organizations.

A simple Single sign-on service was developed for demonstration in this article.

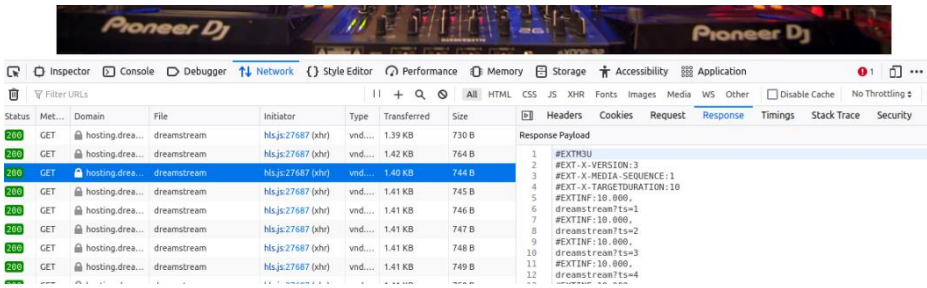

Figure 8: Authorization sessions screen.

Without authentication, access to the video is impossible. Responses to requests are 401 Unauthorized.[7] After invoking the SSO service and successfully passing the authentication process, an application user session is started with a randomly generated cryptographically secure identifier, and the session cookie is returned in the response to the client.

The client is required to submit the token to keep the session active. The time window during which a token is valid, as well as the time for reissuance, are subject to definition on a management level rather than a technical one. The implementation of this SSO service aims to demonstrate that video distribution

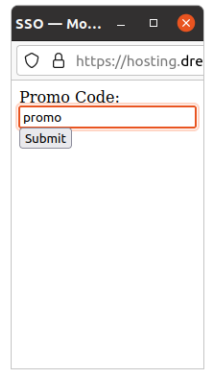

Figure 9: Get authorization token form.

itself and user authentication can be separated between different organizations. This means that the broadcasting organization and the trust provider can be completely separated. Building a streaming platform with a differentiated trust service allows the implementation of various authentication methods that can be performed with:

- user + password
- promo code
- client certificate with corresponding private key on file or smart card
- a mobile application that authenticates users
- through a public platform – email, social network, etc.

Furthermore, their selection or change by the Trust provider at a future time does not require reconfiguration at the video distributor. The second advantage is that the broadcasting organization is disengaged from handling users' personal data.

The downside is that currently, the most common mobile and desktop media players do not support Advanced authentication methods such as SSO, Oauth2, etc., which means that viewing options are limited to a web browser or a specially designed player for mobile and desktop devices. The advantage of media players over browsers is that they provide more control to the user in terms of audio and visual settings, handling the full range of audio and video devices connected to the user's system, etc.

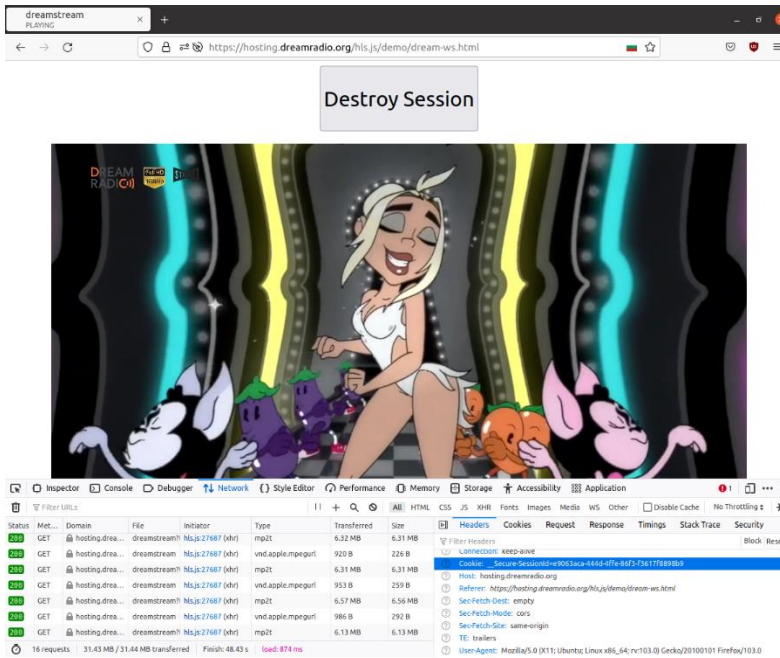

Figure 10: Streaming session authorized.

Conclusions

The research examines the possibilities of protecting the communication between the streaming studio and the RTMP server on the one hand and the HLS communication between the video distributor and the end user, proposing an idea to implement a user authentication portal provided by an authentication service provider that disengages the video distributor from handling personal data and means of payment. For the purpose of the study, the proposed approach was implemented on a test server, from where the results of the study were presented. The advantage of the proposed approach to protecting broadcast streaming content is that it can be applied to any type of high- or low-budget project. It also does not require content distributors to take care of the administration of users' personal data. Which leads to an even higher cyber resilience of the proposed solution, because even with any security breach on a server, there is no personal data that can be stolen.

Acknowledgements

This work has been funded by the National Research Program "Security and Defense" under the framework agreement no. DSD-1/07.07.2022 for the execution of the National Research Program "Security and Defense" between CNSDR-BAS and IICT-BAS, based on the Agreement no. D01-74/19.5.2022

between the Ministry of Education and Science and the Defense Institute for the financing of the research program.

References

1. Documentation Team, "Amazon Kinesis Video Streams Developer Guide," 2018, ISBN-10:9888407848, ISBN-13:978-9888407842.
2. Tasho D. Tashev, Vladimir V. Monov, and Radostina P. Tasheva, "High Performance Computations for Study the Stability of a Numerical Procedure for Crossbar Switch Node," In: Dimov I., Faragó I., Vulkov L. (eds) *Numerical Analysis and Its Applications, 6th International Conference, NAA 2016, Lozenetz,* 2016, LNCS, volume 10187, Springer, Cham, 2017, ISBN:978-3-319-57098-3, DOI:10.1007/978-3-319-57099-0_76, 665-673.
3. Ivan Blagoev, "Neglected Cybersecurity Risks in the Public Internet Hosting Service Providers," *Information & Security: An International Journal* 47, no. 1 (2020): 62-76, https://doi.org/10.11610/isij.4704, https://bpos.bg/publication/18454.
4. Philip Goldie, Matthew Syme, *Optimizing Network Performance with Content Switching: Server, Firewall and Cache Load Balancing* (2003), ISBN: 0131014684.
5. Paul Baka, Jeremy Schatten, Hollie Acres, *SSL/TLS Under Lock and Key: A Guide to Understanding SSL/TLS Cryptography Paperback* (2021), ISBN-10:0648931633, ISBN-13:978-0648931638.
6. Benoit Badrignans, Jean Luc Danger, Viktor Fischer, Guy Gogniat, and Lionel Torres (Eds.), *Security Trends for FPGAS - From Secured to Secure Reconfigurable Systems* (Netherlands: Springer, 2011).
7. John Paul Mueller, *Security for Web Developers: Using JavaScript, HTML, and CSS*, 2015, ISBN-13:978-1491928646.

About the Authors

Iliyan Iliev is a Ph.D. student at the Institute of Information and Communication Technologies – Bulgarian Academy of Sciences. He has an MSc from the University of Library Studies and Information Technologies, and a BSc from the Sofia University St. Kliment Ohridski. Iliyan has strong programming skills and wide system administration experience. The scientific interests of Iliyan are in the fields of computer networks, operating systems, and the Internet of Things. https://orcid.org/0000-0003-3274-9828

Ivan Blagoev, PhD, is an Assistant Profesor in the Modelling and Optimization department in the Institute of ICT in the Bulgarian Academy of Science and a member of research teams exploring digital transformation, cyber security, and cryptography issues. He is involved in projects in the Fin-tech industry, AI, Blockchain, cryptography, and identification. He also has experience as a university lecturer. https://orcid.org/0000-0002-6413-8469