ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 2012. No. 2(118)
ELEKTRONIKA IR ELEKTROTECHNIKA

SYSTEM ENGINEERING, COMPUTER TECHNOLOGY
T 120
SISTEMŲ INŽINERIJA, KOMPIUTERINĖS TECHNOLOGIJOS

Efficient MPEG-2 Transport Stream Encryption Method for Low Processing Power Mobile Devices

V. Simanaitis, A. Liutkevicius, A. Vrubliauskas, E. Kazanavicius
Real Time Computing Systems Centre, Kaunas University of Technology,
Studentų str. 50, LT-51368, Kaunas, Lithuania, e-mails: vytautas.simanaitis@stud.ktu.lt, agnius@ifko.ktu.lt, aras@ifko.ktu.lt, ekaza@ifko.ktu.lt

D. Imbrasas
JSC "Elsis TS",
Uosio St. 10, Kaunas, Lithuania, e-mail: darius.imbrasas@elsis.lt

crossref http://dx.doi.org/10.5755/j01.eee.118.2.1180

Introduction

An intellectual property rights protection became a very common problem in recent years. Especially it is important for the digital video content such as IPTV providers, because their business continuity is directly affected if anybody can use that content illegally. Smart-home and smart-environment content provisioning systems, like SNAPAS [1] also must protect their digital content and ensure, that end-users with their low power terminal devices will be able to watch video stream in real-time. Usually content is protected using digital rights management (DRM) systems, whose allow only authorized users to watch protected content. Various topologies and architectures are used for the IPTV streaming with different DRM solutions, leading to various problems when trying to protect content and still deliver good quality services to customers.

MPEG-4 H.264/AVC video streaming format became very popular recently. It has many advantages comparing with older MPEG H.261 encoding format, because it requires less internet bandwidth, has better decoded video quality, many IPTV STB devices and TV have hardware decoding accelerators.

DRM solutions are using video stream data encryption methods. Usually AES method is used, because it is fast and reliable [2, 3]. Other methods like outdated DES or not very safe XOR are still widely used as well. Not all encryption methods have the same reliability level, while most safe methods are more resource and time consuming. This is not a big problem, when video content users have powerful end-user terminal devices, like high-end PC or similar. The problem arises, when users watch video content using low computing power STB (set-top-box) devices or handheld devices like phones, PDA, etc. These devices are restricted enough and are not able to decrypt streams with complex encryption methods in real-time.

This paper presents novel video content streaming encryption method, which is less computational resource demanding and compatible with end-users video stream decoding systems. This method is suitable for low computing power end-user devices (phones, PDA, STB, etc.) and is capable to encrypt various types of encoded streams, like H.264/AVC or older H.263 and H.261. The proposed encryption method ensures the same level of protection like standard MPEG-4 full encryption method, but it is faster and compatible with almost any existing DRM systems.

Video streaming methods

Video streaming can be categorized into three types: unicast, multicast and broadcast. Paid content is protected from illegal use by adapting data encryption algorithms, while algorithm specific information (like public or private keys, codes, etc.) is provided to the end-users via DRM system using safe channels.

Video streams are of two types: real-time video and already prepared (non real-time) video content streaming. Best known example of video streaming service is internet television (IPTV). Video content should be prepared in such way, that end-users would be able to watch this content using their terminal devices, usually STB connected to TV or TV with integrated STB. IPTV STB device receives video stream from the content provider, decrypts it (if stream is protected), decodes it and sends the output to the TV. At this moment H.264/AVC encoding is most efficient and a best quality ensuring standard.

Both real-time and non real-time streaming uses the same standards and streaming methods. But real-time streaming requires more efficient encoding and encryption methods, because all operations are made on-the-fly, ensuring minimal delays.

Not the all real-time streaming protocols can be applied for different streaming (and protection) methods. Protocols are dedicated for unicast, multicast or broadcast with or without encryption. RTP (Real-Time Transport Protocol) together with SRTP (Secure Real-Time Transport Protocol) are used for the real-time video streaming. SRTP and SRTCP (Secure Real-Time Control Protocol) are the RTP and RTCP (Real Time Control Protocol) versions with data encryption added. SRTP ensures full video stream encryption, while SRTCP is used to share encryption keys and encryption algorithm between content provider and end-user. Other widely used real-time streaming protocol is RTSP (Real Time Streaming Protocol). It is not used for content streaming but rather for controlling streaming servers and providing users with such functions like play, stop, pause, etc. At some cases TCP or HTTP protocols can be used for video streaming as well, but they are not very suitable for such purpose.

A high quality video stream must be compressed effectively in order to send it over a low bandwidth networks. More effective H.264 encoding standard supersedes widely used H.263 encoding standard. H.263 encoding requires 4Mb/s network transfer rate in order to transmit 720x576 25 frames stream, while H.264/AVC encoding requires only 1.5-2 Mb/s transfer rate. H.264 is more effective because of new compression algorithms applied. H.264 encoding is based on video stream frame segmentation in so called macro blocks (MB). According to standard, macro blocks size can be 4x4, 8x8, or 16x16 pixels. There are three types of MB: I-slice, P-slice and B-slice. Each MB is encoded using Discrete Cosine Transform (DCT) and the output is matrix, which is transformed into the digital data sequence using Zig-Zag method. Then the sequence is compressed by Run Length Encoding and Huffman method. Decoding is performed in reverse order and does not require additional computations to divide frames into I, B or P type. H.264 encoding loses some quality because of optimizations during compression, e.g. after applying DCT small coefficients become 0, so before frame is displayed, it should be filtered.

H.264 standard is error-resistant, when errors appear during video stream transmission over unreliable IP networks. The main idea is to find error and to conceal it using appropriate method. Usually it is done by searching similarities in nearby macro blocks, e.g. if 16x16 MB is damaged or lost, it can be treated as zero and displayed as black or green region in the frame. Spatial interpolation, motion compensated interpolation, temporal concealment and similar methods can be applied to repair damaged MB. These methods are good enough to recover one or few MB, but in very noisy channels it is a difficult task, because lost packets contain several MB.

Related work

Compressed video stream is not error resilient, because errors propagate to the subsequent frames and macro blocks. Error propagation can be used for the video stream encryption and protection, because encrypted data are seen as corrupted data at the receiving side. Existing video stream encryption methods and techniques can be categorized into several types as described below.

Full Encryption. There are many video content protection and transmission systems using full stream encryption, like [4] or [5], but they require powerful end-user devices and high bandwidth of the network. Usually full encryption is implemented using SRP and SRTCP protocols [6]. The main advantages of full encryption systems and protocols are independence of video stream encoding standard and very high protection level. The main disadvantage is that decryption process is very resource consuming, hence mobile devices like phones or PDA are not capable to perform such task in real-time.

Selective Encryption. Tang in [7] proposed four different levels of stream encryption with different protection levels. First and least secure method is encryption of all headers. In this case decoder does not know the encoding standard, but such protection is very primitive, since real-time streaming is performed using only few well known standards and headers can be simply guessed. Second level of protection is obtained by encryption of all headers and I-frames. Third level includes encrypting of all I-frames and I-macro blocks in P and B frames, but not the headers. This ensures good protection, because even encoding is known, the content itself is encrypted and cannot be recovered. Fourth level is the most secure, because all frames are encrypted.

Zig-Zag Permutation. This method exploits one of the stages of H.264 encoding algorithm, where 8x8 blocks are transformed into 1x64 sequence using Zig-Zag algorithm. If sequence order is unknown, the decoder cannot recover the matrix and complete decoding successfully. Such method ensures low protection level, because once permutation principle is known, algorithm is not secure any longer. Also such encryption leads to worse compression level of data, because after applying Zig-Zag and DCT algorithms, non-zero coefficients are placed in the beginning of the sequence and can be compressed effectively using Run Length Encoding. In case of Zig-Zag permutation, non-zero coefficients are distributed over the sequence and compression is not so effective [8].

Motion Vector Encryption. Video stream contains frames consisting of macro blocks. Each MB can have motion vector, which specifies which part of frame should be copied into the specific MB. Vector points to the same frame or other frame. MVEA (Motion Vector Encryption Algorithm) was proposed in [9]. Motion vector encryption has two stages. In the first stage motion vectors are concealed using XOR algorithm. In the second stage motion vectors are scrambled with random generated numbers.

DCT coefficient encryption. DCT encryption proposed in [10] is based on three levels of protection. For each level the different number of DCT coefficients is encrypted. First level includes 0-4 coefficients, second 5-19 and third includes remaining 44 coefficients. Depending on desired protection degree, several levels can be encrypted. Coefficients are usually distributed in the first quarter of matrix, which corresponds to the first and

second protection level. Encryption of the first level coefficients guarantees worst protection degree, while encryption of all levels gives the best protection. Such method should maintain a good video stream compression level, because 0 coefficients are not encrypted.

More simple method is proposed in [11], where only signs of DCT matrix coefficients are encrypted. But this encryption is not very strong, since coefficient can have only two possible "+" or "-" signs, which can be guessed. Sign encryption also is used in [12], where signs of motion vectors are encrypted.

Partial encryption. Partial encryption is quite simple, but still effective approach. Partial encryption method of MPEG-2 transport stream is proposed in [13], where transport stream packets containing I-frames are encrypted. But encryption of only I type of blocks does not guarantee good protection level, because some portions of video can be restored from motion vectors. Most of partial encryption methods perform encryption not at the transport level, but at the encoding level, e.g. method proposed in [14] encrypts part of all types of macro blocks using DES.

Analyzed encryption methods have several drawbacks. Some of them require high computational power. Others perform encryption at the encoding level, which gives good protection level, but leads to the incompatibilities with decoding accelerators. Encryption at encoding level also requires re-encoding of at least a half of the video file, when encryption algorithm or secret key needs to be changed. Partial encryption methods are more lightweight and suitable for real-time streaming, but existing techniques are based on the analysis and encryption of specific I, P, or B type of frames or macro blocks, which requires additional computational steps.

In this paper we propose a novel partial encryption method, which performs encryption not at the encoding level, but at the transport level (MPEG-2 transport stream). Proposed method encrypts video stream data not taking into account the type of data (is it motion vector, I-type MB or other type of data). This eliminates additional steps and computations to find I-type macro blocks or motion vectors. Another advantage is that in order to change encryption algorithm or secret key, only transport stream must be re-encrypted, while encoded video data can be prepared before and just reused.

Proposed partial encryption method

Proposed method encrypts MPEG-2 transport stream (which is also used for the MPEG-4 video streaming), that consists of 188 Byte length data packets. Encrypting data at this level should take into account the fact, that H.264/AVC encoding and compression standard has corrupted frames recovery feature, hence enough data should be encrypted to guarantee desired protection level.

The main restriction for the real-time IPTV streaming is a low computing power end-user handheld devices. They are not able do decrypt large portions of video streams encrypted using complex encryption methods. Hence only the most important and hardly guessable parts of video stream should be encrypted. Since STB video decoding is performed using HW, encryption method should not interfere with encoding/decoding process, like many other encryption methods do.

The MPEG-2 transport stream consists of 188B packets, which must be collected in correct order. Otherwise, decoding will not be successful. This means, that mixing the order or simply encrypting the packets, will not allow restoring and decoding the stream, if encryption methods or keys are not known. Proposed method is fully compatible with hardware video decoding, because video stream data would be decrypted before sending them to the decoder. Another advantage is that once video stream is encoded (e.g. using H.264 format), it can be encrypted with different encryption algorithms and keys. This is useful for streaming to the different end-user groups, using different encryption keys.

Proposed partial transport stream encryption method encrypts part of the 188B MPEG-2 transport stream packet. Packet consists of 4B header, and 184B data. For the compatibility with standard purpose the header remains unencrypted and only part of the data payload is encrypted. Moreover, in order to reduce encryption and decryption time and requirements for the decryption device, only part of transport packets are encrypted.

Proposed encryption algorithm consists of several steps:

1. Secure encryption keys are created for the AES algorithm and distributed to the end-users through DRM system.
2. Encryption is based on four parameters, which are set before encryption: *pass_length* defines the length of unencrypted packets block; *crypt_length* defines the length of encrypted packets block, which follows each unencrypted packets block defined by *pass_length*; *pck_offset* defines the encryption offset inside MPEG-2 transport packet's payload data; *pck_crypt_length* defines how much bytes should be encrypted starting from *pck_offset* position.
3. Transport packet is checked and only packet without *adaptation fields* is encrypted.
4. *Pass_length* and *crypt_length* values are evaluated and if packet needs to be encrypted, then part of the packet is encrypted according to the *pck_offset* and *pck_crypt_length* (see Fig. 1). Packet's *scr* flag is set, which means that packet is encrypted (scrambled).
5. Next stream packet is processed (return to step 3).

One of the main advantages of proposed algorithm, comparing with other partial encryption algorithms, is that encryption is performed at the transport level. This means that video content can be encoded using different methods, like H.264, H.263 or H.261, etc. The proposed method is compatible with MPEG streaming standards and no additional data or custom flags are used in transport packet for achieving full compatibility with standards. 188 Byte length transport packet already contains flag, indicating that packet is encrypted.

This flag is reused by proposed encryption algorithm by setting the flag for the encrypted packets.

AES encryption algorithm is used for the experimental evaluation of proposed method. This algorithm is lightweight but still very effective [2, 3].

Any other existing encryption algorithms can be used with proposed encryption method as well.

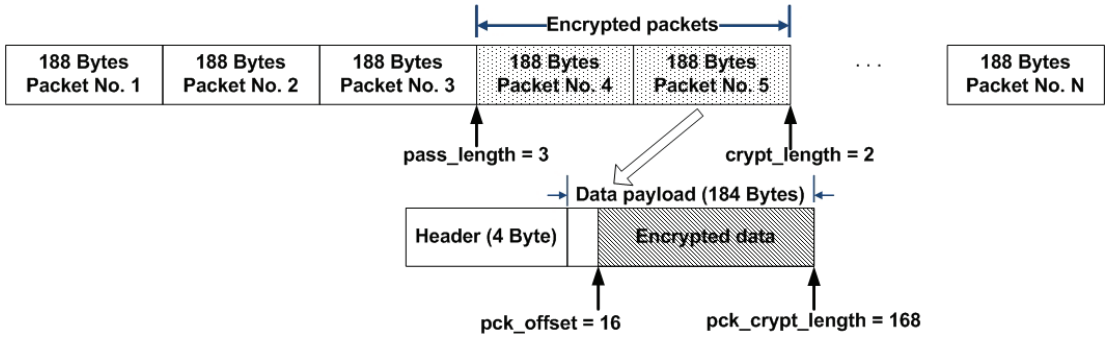


Fig. 1. Proposed partial encryption scheme with example parameters values

Decryption process for the proposed encryption method is rather simple. To decrypt the stream receiver needs to know the secret key, *pck_offset* and *pck_crypt_length* parameters:

1. Received transport packet is inspected for the *scr* flag;
2. If *scr* flag is set, the encrypted part of the packet is identified using *pck_offset* and *pck_crypt_length* parameters and decrypted using secret key;
3. Decrypted packet is sent to decoder;
4. Next stream packet is processed (return to step 1).

Experimental Methodology and Setup

Proposed partial encryption method was evaluated experimentally measuring encryption and decryption times and encrypted video quality. VLC media player (http://www.videolan.org/vlc/) plug-in has been implemented which incorporates proposed encryption algorithm and is able to encrypt and decrypt video data. The plug-in is based on SRTP protocol, where standard encryption is replaced by proposed encryption algorithm.

The main purpose of experimental evaluation was to evaluate the performance of full encryption and proposed protection method at the same time maintaining acceptable protection level.

The encrypted video stream protection level was evaluated measuring video stream quality using peak signal-to-noise ratio (PSNR). Higher PSNR values mean better image quality. For IPTV video streaming at least 30 dB should be reached, while for wireless transmissions 20 dB values are acceptable. If PSNR is below 10 dB, video quality is so low, that end-users see only random noise.

The number of encrypted packets was measured in order to evaluate the percentage of encrypted/decrypted data.

Two experimental video streams were used. One stream contained dynamic video and other stream contained static video (one image shown for several seconds). According to MPEG standard it is obvious, that dynamic videos can be protected with few encrypted packets. Static video needs more encryption, because video can be recovered even from one key frame. We evaluated protection level of proposed method according to the PSNR and visual inspection of the encrypted video.

Intel Atom based end-user low power mobile device was used for the experiments.

VLC plug-in was compiled using *gcc* compiler and integrated with freely available VLC libraries and source code. The plug-in was developed using KDevelop4 with lua language support.

Encryption and decryption times were measured using *perf* tool. Video frames analysis was performed using *pnmpsnr* tool.

Evaluation of Proposed Encryption Method

The evaluation of proposed encryption method was done via series of experiments, with different *pass_length*, *crypt_length*, *pck_offset* and *pck_crypt_length* parameter values. The main purpose was to find the threshold values of the parameters, which guarantee good level of protection with minimum encrypted data.

Table 1 – Table 2 contain encryption and decryption times, encrypted packets numbers, and PSNR values, depending on the *pass_length*, *crypt_length*, *pck_crypt_length, pck_offset* parameters. Experimental evaluation starts with full transport stream packets encryption (*pass_length=0, crypt_length=1*) and ends with *pass_length=24* and *crypt_length=3* values, where group of three packets was encrypted after each 24 unencrypted packets group. At the same time encryption inside the each selected packet was performed with three different combinations of *pck_crypt_length* and *pck_offset* parameters, resulting in 16, 64 or 176 encrypted bytes.

The number of passed and encrypted transport stream packets, encryption and decryption times and PSNR values were found for each combination of *pass_length*, *crypt_length*, *pck_crypt_length, pck_offset* parameters.

In some cases PSNR value was N/A, which means that decoding software was unable successfully decode encrypted stream and simply crashed, which means the highest protection level possible (or like PSNR ≤ 0dB).

The time values presented in Table 1 – Table 2 and Fig. 2 are the average of ten values, i.e. each row of the tables corresponds to the ten experimental measurements.

It can be clearly seen, that even when *pass_length* value is high (24 in our experimental case), the PSNR is still very low (about 10 dB). This means, that somebody who is trying illegally watch this stream see only screen with noise. At the same time the number of encrypted packets and corresponding encryption and decryption times are reduced dramatically comparing with full encryption. Comparing with standard full MPEG encryption algorithm, which encrypts all data payload (184 Bytes) of each stream packet, the proposed encryption method with *pass_length=24, crypt_length=1*, *pck_crypt_length=16* and *pck_offset=168* is about 9 times

faster, but still maintaining almost the same protection level. The last column's first three rows in Table 1 are most close to the standard MPEG encryption with difference only 8 bytes (176 against 184). Full encryption takes ~4.5 s, while 24-1 type encryption takes only ~0.5 s, while decryption takes ~4.2 s and ~0.5 s respectively.

We also performed evaluation of encryption for the static video stream. The results are presented in Table 3 – Table 4 and Fig. 3 – Fig. 6. As mentioned before, it is more difficult to encrypt static images at the transport stream level. The reason is that proposed method does not take into account what type of video frame is inside transport stream packet.

The main feature of static videos is that there is one I-frame with reference image, and other frames are of type B or P. Since proposed method does not analyze frames, additional experimental evaluation was performed to find out the method parameters, which guarantee not only low PSNR value, but also ensure such frequency of encrypted packets, that I-frame becomes encrypted anyway.

Table 1. Encryption performance and quality, encrypted packet count and PSNR values for dynamic video

Encryption								
Sequence of encrypted packets: *pass_length-crypt_length*			**Encryption inside the packet:** *pck_crypt_length* (*pck_offset*)					
	Packets		**16 Bytes (offset 168)**		**64 Bytes (offset 120)**		**176 Bytes (offset 8)**	
	Passed	**Encrypted**	**Time (s)**	**PSNR (dB)**	**Time (s)**	**PSNR (dB)**	**Time (s)**	**PSNR (dB)**
0-1	960	45363	2.8658	2.74	3.2441	2.71	4.4009	N/A
0-2	960	45363	2.7102	2.74	3.1088	2.71	4.3892	N/A
0-3	960	45363	2.5417	2.74	3.0905	2.71	4.2509	N/A
3-1	35000	11323	0.9774	10.25	1.1299	10.12	1.4309	N/A
3-2	28179	18144	1.2897	2.8	1.4817	10.29	1.9756	N/A
3-3	23602	22721	1.5773	2.76	1.7740	2.8	2.3702	N/A
6-1	39845	6478	0.8468	10.26	0.8123	10.12	1.0061	10.24
6-2	35008	11315	0.9691	10.23	1.0951	10.2	1.5107	10.19
6-3	31193	15130	1.1536	4.64	1.3351	2.87	1.6998	N/A
24-1	44509	1814	0.5126	10.08	0.5474	10.08	0.5976	10.26
24-2	42826	3497	0.5871	2.83	0.6658	2.89	0.7770	N/A
24-3	41289	5034	0.6516	10.09	0.7489	10.07	0.8763	N/A

Table 2. Decryption performance and quality, decrypted packet count and PSNR values for dynamic video

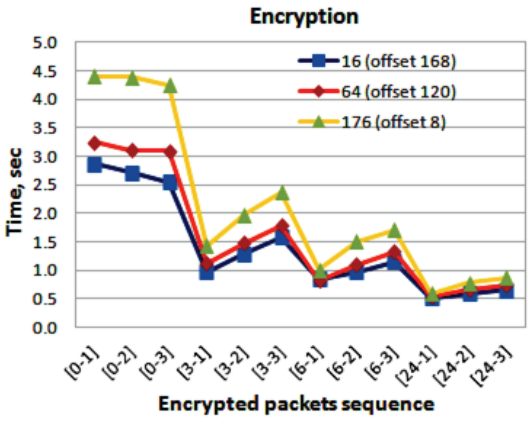
Decryption								
Sequence of encrypted packets: *pass_length-crypt_length*			**Encryption inside the packet:** *pck_crypt_length* (*pck_offset*)					
	Packets		**16 Bytes (offset 168)**		**64 Bytes (offset 120)**		**176 Bytes (offset 8)**	
	Passed	**Decrypted**	**Time (s)**	**PSNR (dB)**	**Time (s)**	**PSNR (dB)**	**Time (s)**	**PSNR (dB)**
0-1	960	45363	2.4868	∞	3.0832	∞	4.1748	∞
0-2	960	45363	2.5308	∞	3.0077	∞	4.1784	∞
0-3	960	45363	2.5175	∞	3.0485	∞	4.3418	∞
3-1	35000	11323	0.9406	∞	1.0720	∞	1.3805	∞
3-2	28179	18144	1.2459	∞	1.4632	∞	1.9340	∞
3-3	23602	22721	1.4841	∞	1.7028	∞	2.5169	∞
6-1	39845	6478	0.7268	∞	0.8115	∞	0.9923	∞
6-2	35008	11315	0.9175	∞	1.0683	∞	1.3605	∞
6-3	31193	15130	1.2854	∞	1.3010	∞	1.6937	∞
24-1	44509	1814	0.5111	∞	0.5253	∞	0.5732	∞
24-2	42826	3497	0.5889	∞	0.6380	∞	0.7115	∞
24-3	41289	5034	0.6643	∞	0.7219	∞	0.8292	∞

Table 3. Encryption performance and quality, encrypted packet count and PSNR values for static video

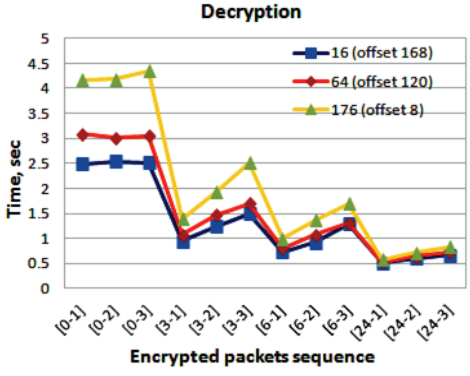
Encryption								
Sequence of encrypted packets: *pass_length-crypt_length*			**Encryption inside the packet:** *pck_crypt_length* (*pck_offset*)					
	Packets		**16 Bytes (offset 168)**		**64 Bytes (offset 120)**		**176 Bytes (offset 8)**	
	Passed	**Encrypted**	**Time (s)**	**PSNR (dB)**	**Time (s)**	**PSNR (dB)**	**Time (s)**	**PSNR (dB)**
0-1	960	45363	0.4091	9.53	0.3093	9.51	0.3731	N/A
0-2	960	45363	0.2491	9.53	0.3022	9.51	0.3939	N/A
0-3	960	45363	0.2525	9.53	0.2921	9.51	0.6663	N/A
3-1	35000	11323	0.1345	7.45	0.1455	7.29	0.1652	8.74
3-2	28179	18144	0.1586	6.98	0.1994	6.98	0.2119	6.98
3-3	23602	22721	0.3487	9.06	0.4034	7.94	0.2389	7.42
6-1	39845	6478	0.1188	11.56	0.1240	11.05	0.1539	9.33
6-2	35008	11315	0.1386	8.88	0.1551	7.86	0.1662	7.84
6-3	31193	15130	0.1761	8.32	0.1610	8.32	0.2033	8.32
24-1	44509	1814	0.1020	34.09	0.1039	45.27	0.1125	45.29
24-2	42826	3497	0.1084	27.15	0.1118	27.15	0.1187	26.62
24-3	41289	5034	0.1269	23.43	0.1219	30.46	0.1444	26.17

Table 4. Decryption performance and quality, decrypted packet count and PSNR values for static video

Decryption								
Sequence of encrypted packets: *pass_length-crypt_length*			**Encryption inside the packet:** *pck_crypt_length* (*pck_offset*)					
	Packets		**16 Bytes (offset 168)**		**64 Bytes (offset 120)**		**176 Bytes (offset 8)**	
	Passed	**Decrypted**	**Time (s)**	**PSNR (dB)**	**Time (s)**	**PSNR (dB)**	**Time (s)**	**PSNR (dB)**
0-1	960	45363	0.2487	∞	0.4146	∞	0.3670	∞
0-2	960	45363	0.2669	∞	0.2993	∞	0.3708	∞
0-3	960	45363	0.2479	∞	0.2815	∞	0.4437	∞
3-1	35000	11323	0.1325	∞	0.1411	∞	0.1623	∞
3-2	28179	18144	0.1723	∞	0.1709	∞	0.2209	∞
3-3	23602	22721	0.3727	∞	0.1927	∞	0.2495	∞
6-1	39845	6478	0.1311	∞	0.1263	∞	0.1447	∞
6-2	35008	11315	0.1331	∞	0.1432	∞	0.1829	∞
6-3	31193	15130	0.1492	∞	0.1583	∞	0.2169	∞
24-1	44509	1814	0.1108	∞	0.1023	∞	0.1067	∞
24-2	42826	3497	0.1079	∞	0.1206	∞	0.1167	∞
24-3	41289	5034	0.1156	∞	0.1313	∞	0.1401	∞

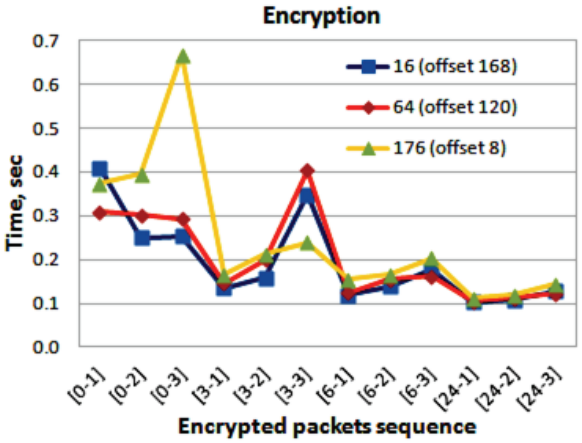


a)

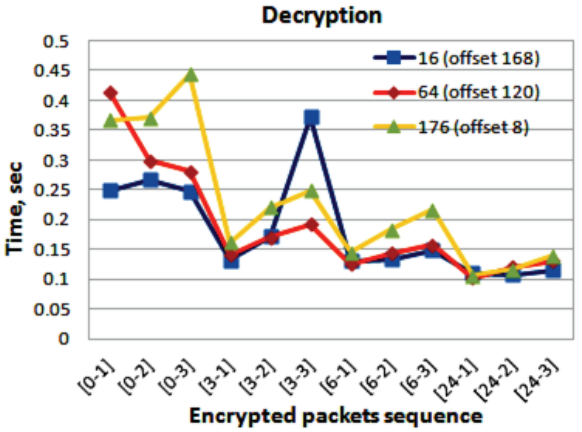


b)

Fig. 2. Encryption and decryption processing times for dynamic video



a)



b)

Fig. 3. Encryption and decryption processing times for static video

We found, that static video needs more frequent packet encryption in order to reduce PSNR values enough. When *pass_length* value is 24, video output is seen as almost unchanged original picture (Fig. 4).

Accordingly PSNR values are very high (~23–45 dB) which also corresponds to acceptable and even very good image quality. PSNR values drop to the acceptable level for protection (~8-11 dB), when *pass_length* ≤ 6.

Visual inspection of such encrypted video shows mainly noise with some color regions, but the original image is unpredictable (Fig. 5 and Fig. 6).

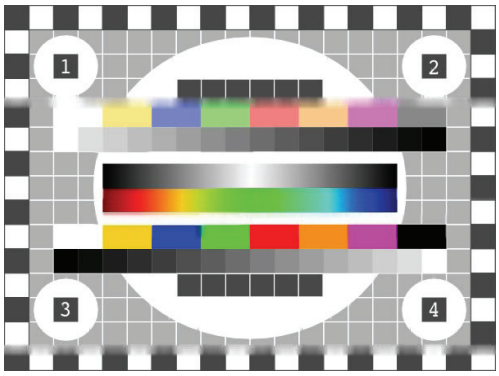


Fig. 4. Static video encrypted with pass_length=24, crypt_length=3, pck_crypt_length=176, pck_offset=8

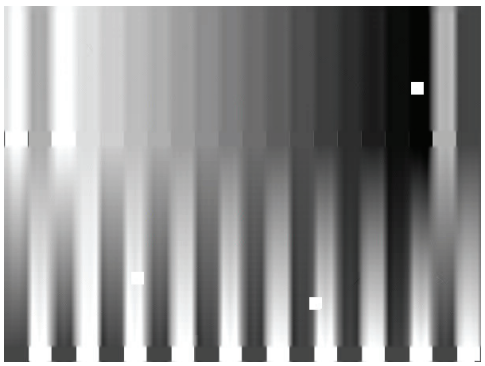


Fig. 5. Static video encrypted with pass_length=6, crypt_length=3, pck_crypt_length=176, pck_offset=8

Fig. 6. Static video encrypted with pass_length=6, crypt_length=3, pck_crypt_length=176, pck_offset=8

Conclusions

In this paper we proposed partial encryption method for MPEG-2 transport stream encryption.

This method is fast and efficient, suitable for end-user's low computing power mobile devices. Only part of transport stream packets is encrypted, encrypting only part of selected packets.

The experimental results show that proposed transport stream partial encryption method ensures good video content protection levels, at the same time substantially reducing encryption and decryption times comparing with standard MPEG full encryption. Method is lightweight and does not require high computing power.

It was found, that for dynamic videos (rapidly changing video frames), it is enough to encrypt 16 Bytes portion of each 24th transport packet, to achieve good protection level (PSNR ~10 dB). For static videos (video frames do not change over time, or changes are minimal) it is enough to encrypt 16 Bytes portion of each 6th transport packet, to achieve good protection level (PSNR ~11 dB). Encryption and decryption times, comparing with full encryption, are reduced from 3 (static video) up to 9 times (dynamic video), making proposed method a good choice for low processing power decryption devices.

References

1. **Liutkevičius A., Vrubliauskas A., Kazanavičius E., Imbrasas D.** Smart home services development, provisioning, and management framework // Information technology and control. – Kaunas: Technologija, 2011. – Vol. 40. – No. 2. – P. 163–169.

2. **Toldinas J., Štuikys V., Ziberkas G., Naunikas D.** Power Awareness Experiment for Crypto Service–Based Algorithms // Electronics and Electrical Engineering. – Kaunas: Technologija, 2010. – No. 5(101). – P. 57–62.
3. **Toldinas J., Stuikys V., Damasevicius R., Ziberkas G., Banionis M.** Energy Efficiency Comparison with Cipher Strength of AES and Rijndael Cryptographic Algorithms in Mobile Devices // Electronics and Electrical Engineering. – Kaunas: Technologija, 2011. – No. 2(108). – P. 11–14.
4. **Shin–Ho L., Han–Yen Y., Jia–Yen W., Jiann–Jone C., Jun–Lin L., De–Hui S.** A Secured Video Streaming System // International Conference on System Science and Engineering, 2010. – P. 625–630.
5. **Nishimoto Y., Imaizumi H., Mita N.** Integrated Digital Rights Management for Mobile TV Using Broadcasting and Communications // 14th Asia–Pacific Conference on Communications (APCC'2008), 2008. – P. 1–5.
6. **Khalifa N. E.–D. M., Elmahdy H. N.** The Impact of Frame Rate on Securing Real Time Transmission of Video over IP Networks // International Conference on Networking and Media Convergence (ICNM'2009). – P. 57–63.
7. **Tang L.** Methods for Encrypting and Decrypting MPEG Video Data Efficiently // Proceedings of the ACM Multimedia, 1996. –P. 219–229.
8. **Qiao L., Nahrstedt K.** Comparison of MPEG Encryption Algorithms // Computers & Graphics. 1998. – Vol. 22. – Iss. 4. – P. 437–448.
9. **Zheng L., Xue L.** Motion Vector Encryption in Multimedia Streaming // Proceedings of the 10th International Multimedia Modelling Conference, 2004. – P. 64–71.
10. **Tosun A. S., Feng W. C.** Efficient Multi–layer Coding and Encryption of MPEG Video Streams // IEEE International Conference on Multimedia and Expo (ICME'2000). – Vol. 1. – P. 119–122.
11. **Shiguo L., Zhongxuan L., Zhen R., Haila W.** Secure Advanced Video Coding Based on Selective Encryption Algorithms // IEEE Transactions on Consumer Electronics, 2006. – Vol. 52. – Iss. 2. – P. 621–629.
12. **Shiguo L., Jinsheng S., Zhiquan W., Yuewei D.** A Fast Video Encryption Scheme Based–on Chaos // 8th International Control, Automation, Robotics and Vision Conference, 2004. – Vol. 1. – P. 126–131.
13. **Jeong–Hyun K., Yeon–Jeong J., Ki–Song Y.** Protection scheme for secure MPEG–2 streaming // IEEE International Conference on Multimedia and Expo, 2004. – Vol. 2. – P. 927–930.
14. **Gunhce K., Dongkyoo S.** Dongil S. Intellectual property management on MPEG–4 video for hand–held device and mobile video streaming service // IEEE Transactions on Consumer Electronics, 2005. – Vol. 51. – Iss. 1. – P. 139–143.

Received 2011 09 12
Accepted after revision 2011 12 05

V. Simanaitis, A. Liutkevicius, A. Vrubliauskas, E. Kazanavicius, D. Imbrasas. Efficient MPEG-2 Transport Stream Encryption Method for Low Processing Power Mobile Devices // Electronics and Electrical Engineering. – Kaunas: Technologija, 2012. – No. 2(118). – P. 81–88.

Video stream encryption is the main method for the protection of intellectual property in modern digital rights management systems (DRM). Existing encryption methods ensure very high protection level of encrypted content, but at the same time are very resources demanding, making them hardly suitable for low processing power mobile end-user devices. This paper presents partial MPEG-2 transport stream encryption method, which is suitable for such low power user terminals. The experimental evaluation of proposed method show that encryption and decryption times, comparing with standard full encryption, are reduced from 3 (static video) up to 9 times (dynamic video), making proposed method a good choice for low processing power decryption devices. Ill. 6, bibl. 14, tabl. 4 (in English; abstracts in English and Lithuanian).

V. Simanaitis, A. Liutkevičius, A. Vrubliauskas, E. Kazanavičius, D. Imbrasas. Efektyvus MPEG-2 transporto srauto šifravimo metodas ribotiems mobiliųjų įrenginių ištekliams skaičiuoti // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2012. – Nr. 2(118). – P. 81–88.

Videosrauto šifravimas yra pagrindinis metodas intelektinei nuosavybei užtikrinti šiuolaikinėse skaitmeninių teisių valdymo sistemose (DRM). Esami šifravimo metodai garantuoja labai aukštą apsaugos lygį, bet tuo pat metu reikalauja labai didelių skaičiavimo išteklių ir yra sunkiai pritaikomi ribotų skaičiavimo išteklių mobiliesiems vartotojų įrenginiams. Šiame straipsnyje pristatomas dalinis MPEG-2 transporto srauto šifravimo metodas, kuris yra tinkamas tokiems ribotų resursų terminaliniams vartotojų įrenginiams. Siūlomo metodo eksperimentinio įvertinimo rezultatai rodo, kad šifravimo ir dešifravimo trukmės, palyginti su standartiniu visišku šifravimu, yra sutrumpinamos nuo trijų (esant statiniam vaizdui) iki devynių kartų (esant dinaminiam vaizdui), todėl šis metodas tinka ribotų skaičiavimo išteklių dešifravimo įrenginiams. Il. 6, bibl. 14, lent. 4 (anglų kalba; santraukos anglų ir lietuvių k.).