Matúš Jókay; Ján Baroš

On the suitability of the internet multimedia storage for steganographic information transfer in MP4 files

Kybernetika, Vol. 48 (2012), No. 3, 522--535

Persistent URL: http://dml.cz/dmlcz/142954

KYBERNETIKA — VOLUME *48* (2012), NUMBER 3, PAGES 522–535

ON THE SUITABILITY OF THE INTERNET MULTIMEDIA STORAGE FOR STEGANOGRAPHIC INFORMATION TRANSFER IN MP4 FILES

Matúš Jókay and Ján Baroš

The aim of this work is to analyze suitability of existing internet multimedia storage services to use as a covert (steganographic) transmission channel. After general overview we focus specifically on the YouTube service. In particular, we study the feasibility of the recently proposed new steganographic technique [6] of hiding information directly in the structure of the mp4-encoded video file. Our statistical analysis of the set of 1000 video files stored by this service show the practical limitations for this type of information hiding.

Keywords: steganography, mp4 file, multimedia storage, YouTube

Classification: 93E12, 62A10

1. INTRODUCTION

A steganographic system uses a communication channel with redundant information (a carrier) to hide a (secret) message. The main difference with classical encryption is that even if the message is encrypted, its existence must still stay hidden. Thus, even if the classical encryption is broken, e. g. by the rapidly progressing new methods of algebraic cryptanalysis [18], the steganographic system can stay useful as a covert channel.

The aim of the steganographic system is to modify the carrier in such a way that any changes made to the carrier are not detectable by (statistical) analysis of the carrier. The most common carrier is a static image. Static image steganography is well-researched from the theoretical point of view. Even more information the static image case can be used in a motion image – video – carrier. However, this area is not yet so well-covered [15]. It is imperative to extend the methodologies from the static, as well as to build new specific ones for the motion case [1]. Even basic definitions, and techniques are not yet thoroughly covered [2].

The most important technique in the static image case is the least significant bit modification (LSB) technique [7]. Its many variants are applied to create different algorithms suitable for specific image encodings. LSB technique can be used both in spatial and frequency domain [11], respectively. Its main advantage is that it can maximize the amount of hidden information in the (fixed size) carrier.

It is possible to extend LSB technique into motion case in a straightforward manner. Our goal is however slightly different. We examine the new steganographic method that

uses directly a specific data structures connected to a flow of data in the video (sequence of images as well as audio information). Although this special channel provides less storage space than LSBs in the contents of the video frames, it can be used to hide specific control information (on a higher security level, if correctly applied).

Only recently a transfer of video files was a difficult technical problem; moreover, each larger data transfer was quickly marked as suspicious. However, nowadays there is a large number of public multimedia storage services, and their use is so common that it does not cause a suspicion. This is a necessary condition for a successful practical steganographic system. Public internet services that provide multimedia storage, such as YouTube, provide also an ideal carrier to mask hidden data transfers in video streams. We examine the current possibilities of such carriers in Section 2.

We proposed a new system based on the structure of mp4 files in [6]. We summarize this system along with basic preliminaries about mp4 file containers in Section 3.

In Section 4 we analyze the characteristics of selected 1 000 mp4 files downloaded from the YouTube service with focus on the practical implementation possibilities of the proposed steganographic system.

2. PUBLIC INTERNET MULTIMEDIA CONTENT PROVIDERS

2.1. Overview of the multimedia storage providers

Public multimedia content storage services are growing ever more popular, and easier to use for a wide range of users. Coupled with the camera enabled mobile phones, the video content generated by the internet users is growing rapidly. Traditional methods of video distribution, such as amateur DVD creation, are unsuitable for most of this type of (ephemeral) content. On the other hand, wide-band connections provide a convenient way to upload and redownload a large number of (relatively) small video-clips. Most of the people however do not have their own web-servers, and this generates a demand for public video hosting services. One of the most popular examples is the YouTube service [16].

We supposed that most of the video services can be used as a hidden channel. However, as shown later, there are some practical limitations, that disallow us to use e. g. YouTube service in practise. Still, it is possible to use it as a side channel, similar to the proposed method in [13].

There is a large number of video hosting services on the internet. We summarize a brief overview of the active providers (in the time of writing). From the steganographic point of view, we are mostly interested in these parameters: maximum video file size, maximum duration, resolution, and the supported file formats.

We compare 20 most visited services (according to Wikipedia [14]). The services and their parameters are summarized in Table 1, 2, and 3, respectively. In Table 1 we present the basic information about the type of content player, and basic video properties. We can see that the most common player is Flash player, but the largest companies also support a new HTML5 standard. The Microsoft Windows media player, and Apple QuickTime are supported only occasionally. Maximum file size is never below 100MB. Some servers even do not limit the file size at all. These services are however strongly specialized (Openfilm provides services for semiprofessional movie makers, Internet Archive

Service	Player type	Max. size (MB)	Max. duration (min)	Resolution
Blip.tv	Flash	1024	Non limited	
Break.com	Flash	500		464x352
Clipshack	Flash	100	Non limited	
Dailymotion	HTML 5, Flash	2048	20	1280x720
Dotsub.com	Flash	700		
flickr.com/explore/video/	Flash	150	1,5	1,280x720
Glumbert	Flash			
iFilm Spike.com	Flash	100		
Internet Archive	HTML 5, Flash	Non limited	Non limited	Non limited
Metacafe	Flash	100		
Multiply		100		
Openfilm	Flash	Non limited	Non limited	690x518
Phanfare		2048	10	
Putfile	Windows Media	25		
Qik	Flash	Non limited	Non limited	640X480
Revver	Apple Quicktime, Flash	100	Non limited	480x392
RuTube	Flash	300		
Sevenload	Flash	1500		1280x720
Tudou	Flash	500	Non limited	
Veoh	Flash	Non limited	30	540x304
Viddix	Flash	250		
Video.aol.com	Flash			
Vimeo	HTML 5, Flash	500	Non limited	1280x720
Yahoo Video	Flash	150		
YouTube	HTML 5, Flash	2048	15	4096x3072

Tab. 1. List of the most populated internet providers of the multimedia storage service.

is not primarily a multimedia storage provider). Maximum duration of the videos is limited only on some servers, but the most popular ones fall into this category (e. g. Dailymotion, YouTube or Flickr). Largest supported resolution varies widely. We note that YouTube supports even the so-called 4k resolution (4096 x 3072 pixels), which is especially suitable to embed a large number of steganographic information in individual frames.

Table 2 summarizes the supported video formats. Almost every service supports 3GP file format that is used in mobile phones, as well as MPEG, which is the most common format used nowadays. Some specific services have only a limited selection of supported format, such as Qik, that is used to share and view video files directly on mobile phones.

Table 3 contains characteristics for two supported multimedia containers (flv and mp4). In the FLV container, the video is encoded using the AVC (Advanced video coding), H.263, and VP6 standards, respectively. The MP4 container supports only AVC.

Služba	MPEG	MOV	WMV	AVI	MKV	MP4	MOD	RA	RAM	ASF	Ogg	3GP	QT
Blip.tv	Y	Y	Y	Y		Y	Y	Y	Y	Y	Y	Y	Y
Break.com													
Clipshack	Y	Y	Y	Y		Y						Y	Y
Dailymotion	Y	Y	Y	Y	Y	Y					Y	Y	Y
Dotsub.com	Y	Y	Y	Y	Y	Y						Y	
flickr.com/explore/video/	Y	Y	Y	Y								Y	
Glumbert													
iFilm Spike.com	Y	Y	Y	Y		Y	Y	N	N	Y		Y	
Internet Archive	Y	Y	Y	Y		Y	Y	Y	Y	Y			
Metacafe	Y	Y	Y	Y		Y	N	N	N	N		Y	Y
Multiply	Y	Y	Y	Y		Y	Y	Y	Y	Y		Y	Y
Openfilm	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y
Phanfare													
Putfile	Y	Y	Y	Y		Y	N	N	N	Y			
Qik	N	Y	N	Y		Y	N	N	N	N	N	Y	N
Revver	Y	Y	Y	Y		Y	N	N	Y				
RuTube													
Sevenload	Y	Y	Y	Y		Y	Y	Y	Y	Y	Y		
Tudou	Y	Y	Y	Y	Y	Y				Y	Y	Y	Y
Veoh	Y	Y	Y	Y									
Viddix	Y	Y	Y	Y		Y							
Video.aol.com													
Vimeo	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Yahoo Video	Y	Y	Y	Y						Y		Y	Y
YouTube	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Tab. 2. Formats supported by the internet providers.

Table 4 contains specific limitations of the services (that apply to our research team). In some cases these restrictions made them unusable for our steganographic research. We considered only the services without restrictions. We focused further only on services that support mp4 containers (due to our new steganographic technique, and also due to the expected replacement of the Flash player by the new HTML5 standard). Finally, we settled on deeper analysis of only the YouTube service for the following practical reasons:

- nowadays it is the most popular video-sharing service on the internet, thus there is also a lot of documentation about the service available;
- the service is available world-wide;
- even in case of future changes, we expect that communication and programming interface will remain documented and supported by the community;
- it is very improbable that the service will stop working in the near future.

Service	FLV				MP4			
	Format	FPS	Bitrate (Kbps)	Resolution	Format	FPS	Bitrate (Kbps)	Resolution
Blip.tv	VP6	24	400	1280x720				
Break.com								
Clipshack	H.263	30,3	458	320x240				
Dailymotion	H.263	29		320x180	AVC	Variable	Variable	512x288
Dotsub.com					AVC	Variable	Variable	420x316
Flickr					AVC	29,97	Variable	640x480
Glumbert	VP6	25	131	320x240				
iFilm Spike.com								
Internet Archive								
Metacafe	AVC	30	664	640x360				
Multiply								
Openfilm					AVC	25	Variable	920x518
Phanfare								
Putfile								
Qik								
Revver								
RuTube								
Sevenload	H.263	25	486	300x240				
Tudou								
Veoh								
Viddix	VP6	25	749	432x324				
Video.aol.com	H.263	25	464	468x352				
Vimeo					AVC	25	Variable	1280x720
Yahoo Video	VP6	30	636	400x222				
YouTube	AVC	24	1036	854x480	AVC	Variable	Variable	1920x1080

Tab. 3. Support of the multimedia containers (flv and mp4) by the internet providers.

2.2. Server communication interface

The communication with the server can be divided into two categories: the operations supported by the service provider (as documented in the official documentation [3]), and the unsupported operations. The first category contains actions that can be directly executed on the YouTube page, as well as the ones that are hidden from the end-users, and available only as programming APIs. Namely, these services contain: user identification, video upload, searching in the list of the videos, etc. However, the service that would enable the **download** of the videos is not officially supported. The video file must be obtained by parsing the source code of the rendered web-page.

YouTube API offers a number of tools to communicate with YouTube servers. These APIs are divided to the three categories according to the programming skills of the user. The basic APIs are Custom Player, and Widgets, more advanced APIs contain Player API, and Embedded Player, and finally the most advanced APIs are the Data API and Chromeless Player. On our level of research we used the Data API.

Data API allows the programmer to employ a number of operations available on the YouTube page automatically. It is possible to search for videos, get lists of videos, etc.

Service	Note
Blip.tv	
Break.com	Not possible to download video
Clipshack	
Dailymotion	
Dotsub.com	
Flickr	
Glumbert	
iFilm Spike.com	Not possible to download video
Internet Archive	Is an internet library primary, not a video service
Metacafe	
Multiply	Is a social network primary, not a video service
Openfilm	
Phanfare	Service not for free use
Putfile	Service not available yet
Qik	Service only for mobil devices
Revver	Service not available at the moment
RuTube	Russian API
Sevenload	
Tudou	Chinese API
Veoh	Service not available in Slovakia
Viddix	
Video.aol.com	Service available only in USA
Vimeo	
Yahoo Video	
YouTube	

Tab. 4. List of the suitable internet providers of the multimedia storage service.

Data API also contains user identification functions, upload functions, user preferences functions, etc. Data API is not limited to a single programming language, but it can be used on many platforms, including pure HTML and XML, Java, .NET, PHP and Python.

2.3. YouTube URL structure

The basic URL of the YouTube service has the following format: `http://www.youtube.com/watch?v=abcdefghijk`. It contains a single parameter, which is the video ID (parameter named "v"). It is usually the first parameter in the (GET) URL. It contains 11 characters. The ID of the video can also be obtained using the API (searching and listing).

When creating custom URLs to enable download of the video, we can use "fmt" parameter to specify quality of the downloaded video (encoder parameters). In the present this parameter can contain ten possible values, which specify all combinations of resolutions and formats provided by the YouTube service. These values are summarized in Table 5.

fmt	Container	Video		Audio	
		Coding	Resolution	Coding	Channels
5	FLV	H.263	400x240	MP3	1-2
34		H.264/AVC	640x360	AAC	2
35			854x480		
18	MP4		480x360		
22			1280x720		
37			1920x1080		
38			4096x3072		
43	WebM	VP8	854x480	Vorbis	
45			1280x720		
17	3GP	MPEG4	176x144	AAC	

Tab. 5. Formats and resolutions available from YouTube service.

Finally, we need to specify the parameter "title". This parameter is used as a filename for the video file stored locally. If we do not specify this parameter, video is not downloaded, but it is only played in the embedded player.

3. STEGANOGRAPHIC SYSTEM BASED ON MP4 FILES

3.1. MP4 container

MP4 is a multimedia container used also by the YouTube service to combine audio and video streams into a single file (stream). It is specified in the international standard ISO 14496-14 [9]. It is based on the more general file structure from ISO 14496-12 [8], which was historically based on the container format QuickTime from Apple. The MP4 container usually contains video streams encoded in MPEG-1, MPEG-2, MPEG-4, and AVC formats. Audio streams are encoded in MP3 or AAC formats. Unlike older containers (e. g. AVI) it can also contain special items such as menu, subtitles, more audio tracks etc. Moreover, it also supports 3D objects.

All data stored in MP4 file are organized as a set of so called *atoms*. Atom is a container, which is identified by its *type* and *length* [10]. The structure of atoms inside MP4 file is demonstrated in Table 6 [8]. Not all MP4 files contain all types of atoms. We will focus only on those atoms that are relevant for our steganographic algorithm.

Generally, MP4 file has only three types of atoms on the top level: `moov`, `mdat`, and `free`, respectively. All presentation metadata are stored in the structure, which is stored in the atom called `moov` (from *Movie Box*, the terminology is a legacy from the QuickTime). Only the moov atom has a particular internal structure with special subatoms. Atom type called `free` is used for the *unused part* of the file. Atom type called `mdat` (*Media Data Box*) contains the actual data from the audio and video streams.

`Moov` contains atom with the header, and then several tracks. Movie header (`mvhd` – *Movie Header Box*) provides basic metadata, such as video duration. Atoms of type `trak` (*Track Box*) contains metadata about individual tracks in the file. Similar to the `moov`

moov				Metadata container
	mvhd			Movie header
	iods			Object descriptor
	trak			Stream tracks container
		tkhd		Track header
		tref		Track container
		edts		Edit list container
			elst	Edit list
		mdia		Track media information
			mdhd	Media header
			hdlr	Media type handler
			minf	Media information container
				vmhd Video header (only for video track)
				smhd Audio header (only for audio stream)
				hmhd Header for helper tracks
				<mpeg> MPEG stream header
				dinf Data information container
				dref Data reference atom (holds stream source)
				stbl Samples table
				stts Frames duration
				dtts Decode time of the frames
				stss Map of the I frames
				stsd Frames description
				stsz Frames lenght
				stsc Chunk lenghts
				stco Chunk offsets
				stsh Optional frames
				stdp Priority damaged frames
mdat				Data container
free				Empty space
skip				Empty space
udta				User data, copyright, etc...

Tab. 6. Atoms structure of the mp4 container.

atom, also `trak` atoms contain header: `tkhd` (*Track Header Box*), with track metadata. `Trak` atom then contains several more subatoms that describe how the data are stored in the `mdat` atom, and how should they be accessed. The content type is stored in the `hdlr` (*Handler Reference Box*), containing details on the media type ("`vide`" is abbreviation for video, "`soun`" is for sound). According to media type, `minf` (*Media Information Box*) atom then contains either `vmhd` atom for video or `smhd` atom for audio.

The most interesting for our work are atoms inside the `stbl` (*Sample Table Box*). This is the atom that contains tables with information about individual samples in the `mdat` atom. `Stbl` contains all time and data indices of the samples in video and audio tracks (for the particular `trak` atom that is the parent of a given `stbl` atom). `Sbtl` is physically structured again as a set of atoms, which are again in the form of tables.

Internal tables of `sbtl` can be used to locate samples in time, find their type (whether they are `I` or `P` frames), their size, container, and their place in this container.

`Stts` atom (*Time to Sample* atom) contains timing information (durations of individual samples). Each table line contains a number of samples and their duration. `Stss` atom (*Sync Sample Atom*) stores frame information for the frames that can be directly accessed without the need to decode other frames. `Stsc` atom (*Sample to chunk atom*) contains information about the first chunk from the set of successive chunks with similar properties. Chunk is a sequence of data frames of the same type physically stored in `mdat` atom. The next atom of metadata is `stsz` (*Sample size* atom). It contains the number of samples and a table with sample sizes of individual samples. The last atom mentioned is `stco` (*Chunk offset* atom). The atom contains a table with initial positions of each chunk in the file. These positions are relative to the beginning of the file, not to the beginning of the `mdat` atom. This allows to reference multimedia content without the need to have a specific container encoding. The disadvantage is that the change of atoms before `mdat` can change the positions of chunks, and the header must be recoded. The structure of `mdat` is summarized in Figure 1.

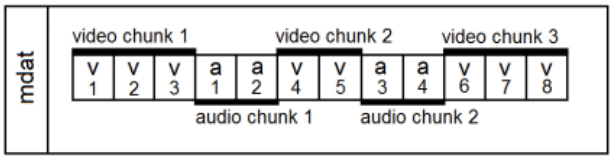

Fig. 1. Alternating of the video and audio streams in the `mdat` atom of the mp4 container.

3.2. Steganographic system using MP4 structures

Proposed steganographic system based on MP4 files [6] uses the physical sequence of chunks in mdat atom to encode the hidden information. As seen on Figure 1, individual data streams use a time multiplex in the physical data stream. In every moment it is possible (in this case we suppose only a single thread is used for video decoding) to read only the data from a single data stream. Using the time multiplex means, that each stream can only occupy a fixed quantum of processing time. In this quantum only a limited set of data frames is processed, and this set is denoted by the term chunk.

The steganographic system uses the fact that encoder standards from the MPEG family do not specify the exact number of data frames in each data chunk in mp4 container. Thus the hidden message can be encoded using the number of frames in subsequent data chunks. The necessary condition for successful employment of this method is that the number of frames varies sufficiently (have enough entropy).

`Mdat` atom contains alternating video and audio chunks. In practice, the number of frames in a chunk is derived from the special parameter of the encoder: *bitrate*.

In Figure 2, we can see the sequence of frame numbers in chunks for a fixed bitrate. Values 15 and 16 form specific pattern (given by the bitrate). A static bitrate usually

15	15	15	15	15	16	15	15	15	15	15	16	15	15	15	15	15	16	15	15	15	15	15	16	15	15	15
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0

Fig. 2. Video chunks example.

leads to only a maximum of two values for video chunk sizes. In a steganographic system these two values can represent zeroes and ones, as depicted in Figure 2.

Steganographic system based on chunk sizes encodes information bits by the number of frames in individual chunks. The change of physical ordering of chunks in `mdat` atom however also leads to recoding of control information and metadata (`stts`, `stss`, `stsc`, `stsz` a `stco` atoms).

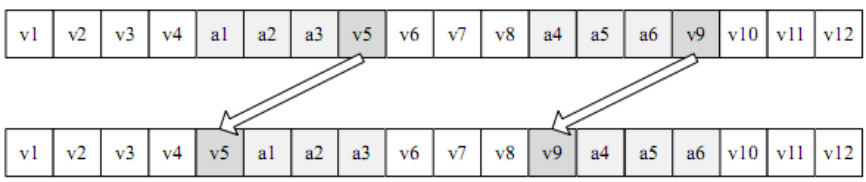

Fig. 3. Information coding example.

This is demonstrated in Figure 3. If we want to insert information bit 0, we can encode it as an even number of frames in a chunk. If the original number of frames is even, nothing gets changed. If it is odd, we must change the number of frames in this chunk. To encode information correctly, and not to disturb previously stored hidden message, we must take a frame from the next chunk and move it to the chunk we are modifying (or we can create a *dummy* chunk). We must then update all affected metadata.

Unfortunately, it is possible to detect dummy chunks very easily. It is thus necessary to consider their usage, e. g. to check whether similar chunks are already present in the original carrier. If not, the system should not use them at all. The second type of problem is the injection of the frame, which references incorrect data in previous frame(s). We need to measure the visual degradation of the resulting image. If the degradation is visually detectable, we need to combine both of the previously mentioned methods: use empty macroblocks along with backward macroblock references.

Note that this technique can be used not only with video stream, but also with other content in mp4 container.

Our encoding does not use video data directly. We have intentionally avoided the modification of the data that can be changed by the used decoder. Current video standards do not specify data types that decoders should use. This allows to speed up the decoding process by using more advanced algorithms, which use integer instead of floating point operations. However, this leads to possible rounding errors, and the loss of the hidden message. We need to carefully study the possible ways how the information might survive, e. g. in ways similar to mobile networks environment [12].

The basic steganographic methods embed directly the provided information without

considering possible encryption. It might be possible to combine steganographic part with lightweight variable-length encryption, based on [4], to enhance the overall security of the system. However, we must then consider an effect of possible combination of automatic cryptanalytic methods, such as [5], with the identification of the hidden message directly in the encrypted text [17].

4. THE EXPERIMENTAL RESULTS

One of the main goals of the steganographic system is undetectability. The proposed steganographic system uses a special physical structure of chunks to encode hidden information. This structure in practice depends on the bitrate parameter of the encoder. We examined 1000 videos from the YouTube service to find whether their structure is suitable for data hiding in the proposed form. However, all videos downloaded have a static bitrate, so that each video file contains only two possible lengths of data chunks.

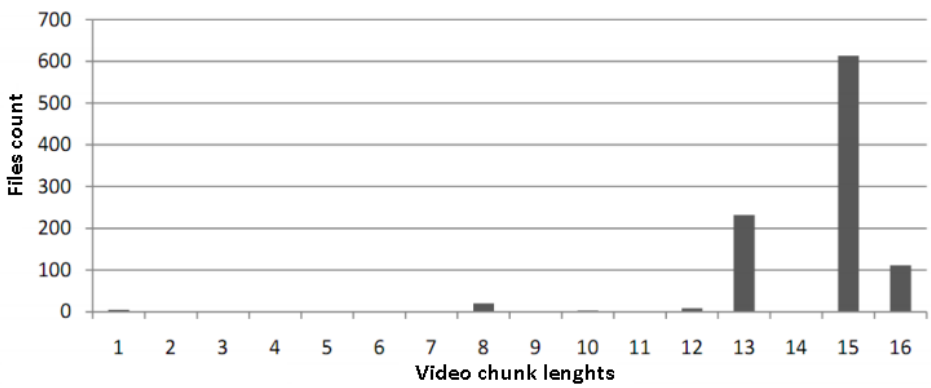

Fig. 4. Downloaded video files classified by the chunk lengths.

Figure 4 contains a distribution of data chunk lengths in the analyzed files. 613 video files had basic chunk length 15, 232 had basic length 13, and 111 had length 16, respectively. 95% of all video files thus had chunk sizes 13, 15 or 16.

We converted each sequence of chunk lengths into binary encoding according to their parity. None of the sequences is uniformly distributed. Histogram of ratios of ones to zeroes in individual files is shown in Figure 5.

Binary sequences corresponding to individual video files can be divided into 33 classes (Figure 6). The sequences are periodic. This is a consequence of the static bitrate. Such non-random sequences are thus unusable for steganographic purpose as designed. The method is only usable with variable bitrate, and as our experiments show, such videos are not common at present. Unless there are more variable bitrate videos in the future, we must conclude that the method as designed does not meet the goals of steganography in practice.

The last goal examined was the robustness of carrier from the steganographic point of view. This means we interested in the fact whether the hidden message remains when

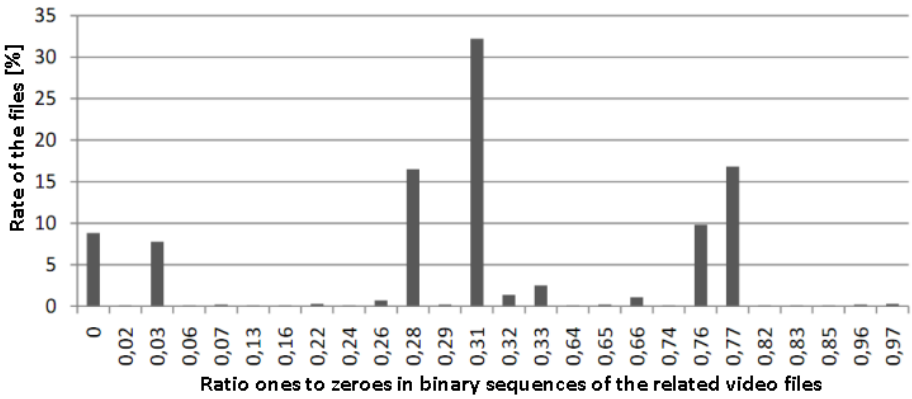

Fig. 5. Distribution of the video files according to ratios of ones to zeroes in related binary sequences.

the video is uploaded to the YouTube. We created a special steganographic video file for the experiment, which copied the most frequent characteristics used by YouTube. Using APIs we uploaded this video to the YouTube storage. After the download, we examined the internal structure of the file. The bitrate parameter is changed by the service, as well as the number of frames in chunk, and other internal structures used in the original proposal [6]. Then we repeated the experiment with the downloaded video. Our results show that even when all parameters where the same, the service always reencodes the video to its internal (undisclosed) standards. Although the changes are visually undetectable, the changes in internal structure cause a destruction of the steganographic information. The YouTube service is thus not suitable for the (proposed kind of) steganography. This might be intentional.

5. CONCLUSIONS

Although the steganographical system [6] based on physical data storage in mp4 files is theoretically possible, its practical realization is limited by the available encoders of the video file. We shown, that in practice, files on YouTube storage are encoded with constant bitrate which is unsuitable for our system. Moreover, each video file is reencoded by the service, removing the steganographic information in the process.

However, the users are not limited to the YouTube service. There are many providers that allow the storage of files in exactly the same format which is chosen by the user. Moreover, it is also possible to use generic data storage services such as RapidShare. In this case it is guaranteed that the content of the files is not modified. On the other hand, from the point of view of law-enforcing agencies, the use of steganographic systems can be undesirable, and it would be advisable to enforce reencoding of any multimedia file with variable bitrate also on generic storage services.

Fig. 6. The classes of the binary sequences related to the examined video files.

ACKNOWLEDGEMENT

This research was supported by the grant VEGA 1/0244/09.

(Received July 29, 2011)

REFERENCES

[1] L. Bielik: The Raven paradox, logic, and methods of testing. Organon F, Internat. J. Anal. Philos. *18* (2011), 2, 213–225.

[2] F. Gahér, L. Bielik, and M. Zouhar: On definitions and defining. Filozofia *65* (2010), 8, 719–737.

[3] Google: API Overview Guide. 2011, `http://code.google.com/apis/youtube/getting_started.html#data_api`.

[4] O. Grošek, P. Horák, and P. Zajac: On complexity of round transformations. Discrete Math. *309* (2009), 18, 5527–5534.

[5] O. Grošek and P. Zajac: Automated cryptanalysis. In: Encyclopedia of Artificial Intelligence (J. R. Rabunal Dopico, J. Dorado De La Calle, and A. Pazos Sierra, eds.), 2009, pp. 179–185.

[6] M. Jókay: The design of a steganographic system based on the internal MP4 file structures. WSEAS Trans. (2011), preprint.

[7] I. Moskowitz, G. Longdon, and L. Chang: A new paradigm hidden in steganography. In: Workshop on new security paradigms (2000). ACM Press, New York, pp. 41–50.

[8] Motion Picture Experts Group: ISO/IEC 14496-12:2004 Information Technology – Coding of Audio-visual Objects – Part 12: ISO Base Media File Format. International Organization for Standardization, 2004.

[9] Motion Picture Experts Group: ISO/IEC 14496-12:2004 Information Technology – Coding of Audio-visual Objects – Part 14: MP4 File Format. International Organization for Standardization, 2004.

[10] F. Pereira C. N. and E. Touradj: The MPEG-4 Book. Upper Saddle River, Prentice Hall PTR, NJ 2002.

[11] K. Solanki, N. Jacobsen, U. Madhow, B. S. Manjunath, and S. Chandrasekaran: Robust image-adaptive data hiding based on erasure and error correction. In: IEEE Trans. Image Process. *14* (2004), 1627–1639.

[12] M. Vojvoda: A survey of security mechanisms in mobile communication systems. Tatra Mt. Math. Publ. *25* (2002), 101–117.

[13] M. Vojvoda and O. Grošek: Postranné kanály v kryptoanalýze. EE časopis pre elektrotechniku a energetiku *14* 14 (2008), 26–29.

[14] Wikipedia: Comparison of Video Services.
2010, `http://en.wikipedia.org/wiki/Comparison_of_video_services`.

[15] C. Xu, X. Ping, and T. Zhang: Steganography in compressed video stream. In: Proc. First Internat. Conference of Innovative Computing, Information and Control (2006).

[16] YouTube: YouTube Homepage. 2011, `http://www.youtube.com`.

[17] P. Zajac: Ciphertext language identification. J. Electr. Engrg. *57* (2006), 7/s, 26–29.

[18] P. Zajac: On the use of the method of syllogisms in algebraic cryptanalysis. In: Proc. 1st Plenary Conference of the NIL-I-004: Bergen 2009, pp. 21–30

Matúš Jókay, Slovak University of Technology, Faculty of Electrical Engineering and Information Technology, Ilkovičova 3, 812 19, Bratislava. Slovak Republic.
e-mail: matus.jokay@stuba.sk

Ján Baroš, Slovak University of Technology, Faculty of Electrical Engineering and Information Technology, Ilkovičova 3, 812 19, Bratislava. Slovak Republic.
e-mail: xbarosj@is.stuba.sk