Trusted Watermarks
*a prototype for a so-called Trusted Set-Top-Box*

Andreas Brett
andreas_brett@ethemba.info

February 23, 2009
This work was supported by

Dr. Andreas U. Schmidt (CREATE-NET)
Nicolai Kuntze (Fraunhofer-SIT)
# Contents

1 apps ............................................ 7
   1.1 AnalyzeWatermark ................................. 8
      1.1.1 Description ................................ 8
      1.1.2 PUBLIC main ................................ 8
   1.2 ImportVideo .................................... 9
      1.2.1 Description ................................ 9
      1.2.2 PUBLIC main ................................ 9
   1.3 PeerListRetrieval ............................... 10
      1.3.1 Description ................................ 10
      1.3.2 CONSTRUCTOR PeerListRetrieval ............... 11
      1.3.3 PUBLIC run ................................ 12
   1.4 Vclient .......................................... 13
      1.4.1 Description ................................ 13
      1.4.2 PUBLIC main ................................ 15
   1.5 VideoChunkRetrieval ......................... 16
      1.5.1 Description ................................ 16
      1.5.2 CONSTRUCTOR VideoChunkRetrieval .......... 17
      1.5.3 PUBLIC run ................................ 17
   1.6 VideoChunkTransmission .................. 18
   1.7 Vserver ......................................... 19

2 modules ........................................ 21
   2.1 FridrichPatternGenerator .................... 22
      2.1.1 Description ................................ 22
      2.1.2 CONSTRUCTOR FridrichPatternGenerator ....... 22
      2.1.3 CONSTRUCTOR FridrichPatternGenerator ....... 23
      2.1.4 PUBLIC get ................................ 23
   2.2 JPEGmarker ..................................... 24
      2.2.1 Description ................................ 24
      2.2.2 CONSTRUCTOR JPEGmarker ................... 25
      2.2.3 CONSTRUCTOR JPEGmarker ................... 25
      2.2.4 PUBLIC embedBits ............................ 25
      2.2.5 PUBLIC readBits ............................. 25
      2.2.6 PUBLIC saveJPEG ............................. 26
      2.2.7 PUBLIC saveJPEG ............................. 26

3 net ............................................. 27
   3.1 NetCommand ................................... 28
      3.1.1 Description ................................ 28
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 types</td>
<td></td>
</tr>
<tr>
<td>4.1 BitString</td>
<td>Description</td>
</tr>
<tr>
<td>4.2 Block</td>
<td>Description</td>
</tr>
<tr>
<td>4.3 BlockVector</td>
<td>Description</td>
</tr>
<tr>
<td>4.4 EmbeddedInformation</td>
<td>Description</td>
</tr>
</tbody>
</table>
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4.3 PUBLIC getMessage</td>
<td>44</td>
</tr>
<tr>
<td>4.4.4 PUBLIC getInformation</td>
<td>44</td>
</tr>
<tr>
<td>4.4.5 PUBLIC getInformation</td>
<td>44</td>
</tr>
<tr>
<td>4.4.6 PUBLIC getFrameNum</td>
<td>45</td>
</tr>
<tr>
<td>4.5 FloatBlock</td>
<td>46</td>
</tr>
<tr>
<td>4.5.1 CONSTRUCTOR FloatBlock</td>
<td>46</td>
</tr>
<tr>
<td>4.5.2 CONSTRUCTOR FloatBlock</td>
<td>46</td>
</tr>
<tr>
<td>4.5.3 CONSTRUCTOR FloatBlock</td>
<td>46</td>
</tr>
<tr>
<td>4.5.4 CONSTRUCTOR FloatBlock</td>
<td>46</td>
</tr>
<tr>
<td>4.5.5 PUBLIC doDCT</td>
<td>47</td>
</tr>
<tr>
<td>4.5.6 PUBLIC doIDCT</td>
<td>47</td>
</tr>
<tr>
<td>4.5.7 PUBLIC toIntBlock</td>
<td>48</td>
</tr>
<tr>
<td>4.6 FloatBlockVector</td>
<td>49</td>
</tr>
<tr>
<td>4.6.1 Description</td>
<td>49</td>
</tr>
<tr>
<td>4.6.2 CONSTRUCTOR FloatBlockVector</td>
<td>49</td>
</tr>
<tr>
<td>4.6.3 CONSTRUCTOR FloatBlockVector</td>
<td>49</td>
</tr>
<tr>
<td>4.6.4 CONSTRUCTOR FloatBlockVector</td>
<td>49</td>
</tr>
<tr>
<td>4.7 IntBlock</td>
<td>50</td>
</tr>
<tr>
<td>4.7.1 Description</td>
<td>50</td>
</tr>
<tr>
<td>4.7.2 CONSTRUCTOR IntBlock</td>
<td>50</td>
</tr>
<tr>
<td>4.7.3 CONSTRUCTOR IntBlock</td>
<td>50</td>
</tr>
<tr>
<td>4.7.4 CONSTRUCTOR IntBlock</td>
<td>50</td>
</tr>
<tr>
<td>4.7.5 PUBLIC doDCT</td>
<td>51</td>
</tr>
<tr>
<td>4.8 IntBlockVector</td>
<td>52</td>
</tr>
<tr>
<td>4.8.1 Description</td>
<td>52</td>
</tr>
<tr>
<td>4.8.2 CONSTRUCTOR IntBlockVector</td>
<td>52</td>
</tr>
<tr>
<td>4.8.3 CONSTRUCTOR IntBlockVector</td>
<td>52</td>
</tr>
<tr>
<td>4.8.4 CONSTRUCTOR IntBlockVector</td>
<td>52</td>
</tr>
<tr>
<td>4.9 JPEGimage</td>
<td>53</td>
</tr>
<tr>
<td>4.9.1 Description</td>
<td>53</td>
</tr>
<tr>
<td>4.9.2 CONSTRUCTOR JPEGimage</td>
<td>53</td>
</tr>
<tr>
<td>4.9.3 CONSTRUCTOR JPEGimage</td>
<td>53</td>
</tr>
<tr>
<td>4.9.5 PUBLIC embedWatermark</td>
<td>54</td>
</tr>
<tr>
<td>4.9.6 PUBLIC readWatermark</td>
<td>54</td>
</tr>
<tr>
<td>4.9.7 PUBLIC save</td>
<td>54</td>
</tr>
<tr>
<td>4.9.8 PUBLIC save</td>
<td>55</td>
</tr>
<tr>
<td>4.10 MJPEGmovie</td>
<td>56</td>
</tr>
<tr>
<td>4.10.1 Description</td>
<td>56</td>
</tr>
<tr>
<td>4.10.2 CONSTRUCTOR MJPEGmovie</td>
<td>56</td>
</tr>
<tr>
<td>4.10.3 PUBLIC getFrame</td>
<td>56</td>
</tr>
<tr>
<td>4.10.4 PUBLIC numFrames</td>
<td>56</td>
</tr>
<tr>
<td>4.10.5 PUBLIC close</td>
<td>57</td>
</tr>
<tr>
<td>4.10.6 PUBLIC readWatermark</td>
<td>57</td>
</tr>
<tr>
<td>4.10.7 PUBLIC convertJPEGtoMJPEG</td>
<td>57</td>
</tr>
<tr>
<td>4.10.8 PUBLIC convertMJPEGtoJPEG</td>
<td>58</td>
</tr>
<tr>
<td>4.10.9 PUBLIC markDirectory</td>
<td>58</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>4.10.10</td>
<td>PUBLIC encryptDirectory</td>
</tr>
<tr>
<td>4.10.11</td>
<td>PUBLIC decryptDirectory</td>
</tr>
<tr>
<td>4.11</td>
<td>PeerList</td>
</tr>
<tr>
<td>5</td>
<td>utils</td>
</tr>
<tr>
<td>5.1</td>
<td>Colors</td>
</tr>
<tr>
<td>5.2</td>
<td>Images</td>
</tr>
<tr>
<td>5.3</td>
<td>PRNG</td>
</tr>
<tr>
<td>6</td>
<td>Settings</td>
</tr>
<tr>
<td></td>
<td>References</td>
</tr>
</tbody>
</table>
This package contains applications implementing several protocols used by Trusted Set-Top-Boxes.

- **AnalyzeWatermark** helps analyzing watermarked media (videos and images) and may be used to verify successful watermarking of a video
- **ImportVideo** permits importing and pre-processing MJPEG videos
- **PeerListRetrieval** and **Vserver** implement the client and server side of the **PeerListRetrieval** protocol
- **VideoChunkRetrieval** and **VideoChunkTransmission** implement the client and server side of the **VideoChunkRetrieval** protocol
- **Vclient** is a demonstrator implementing a complete download and watermarking process for a **Trusted Set-Top-Box**
1.1 AnalyzeWatermark

1.1.1 Description

AnalyzeWatermark is a command-line wrapper for MJPEGmovie and JPEGmarker. It allows for reading out watermarks from whole MJPEG-videos or single JPEG-images.

The watermark seed used for reading out the embedded message is obtained from Settings and thus is constant. AnalyzeWatermark may accordingly be used to verify watermarked videos created by a Trusted Set-Top-Box, as those (for demonstration purposes) are watermarked using a constant watermark seed defined in Settings.

1.1.2 PUBLIC main

Description
Called when used in command-line.

Settings
sequenceBitsPerFrame, payloadBitsPerFrame, Vserver_defaultWMseed

Parameters

<table>
<thead>
<tr>
<th>STRING[]</th>
<th>args</th>
<th>[1] media type ('mjpeg' or 'jpeg')</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>[2] file name of video/image</td>
</tr>
</tbody>
</table>

Output
If no command line parameters are given, a usage information is displayed.
1.2 ImportVideo

1.2.1 Description

ImportVideo allows for importing a MJPEG video to the Vserver. The video is stored under a given video name and converted into AES-encrypted chunks. These are being generated by extracting each frame from the video (using `PUBLIC convertMJPEGtoJPEG`) and subsequently encrypting them (using `PUBLIC encryptDirectory`).

The initial seed used for AES is stored on the disk together with an initial and empty `PeerList`. The name and location of the video as well as the output directory for storing chunks, AES-seed and PeerList in are supplied as command-line parameters. For convenience and tests, the switch ‘/f’ provides a fixed mode, reading the globally set values for this purpose.

1.2.2 `PUBLIC` main

**Description**
Called when used in command-line.

**Settings**
Import_defaultVideoName, Import_defaultVideoFile, Import_defaultOutputDir

**Parameters**

<table>
<thead>
<tr>
<th>STRING[]</th>
<th>args</th>
<th>[1] video name or ‘/f’ for fixed mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[2]</td>
<td>video file</td>
</tr>
<tr>
<td></td>
<td>[3]</td>
<td>output directory</td>
</tr>
</tbody>
</table>

**Output**
If no command line parameters are given, a usage information is displayed.
1.3 PeerListRetrieval

PeerListRetrieval is the client-side implementation of the **PeerListRetrieval** protocol which is used between a Trusted Set-Top-Box and the Video-Server. It is used by a TSTB to request essential data for downloading a video.

Figure 1.1: Protocol: PeerListRetrieval

1.3.1 Description
This essential data includes:

1. aesSeed: seed used to derive the AES key the video was encrypted with
2. wmSeed: seed used to base watermark operations on
3. PeerList: a **PeerList** containing peer ⇔ chunks mappings
4. EmbeddedInformation: an **EmbeddedInformation** object containing a pre-processed message to be embedded into the video

To obtain this dataset from a Video-Server, a Trusted Set-Top-Box has to prove being unaltered by presenting a RA_certificate which was previously obtained in a Remote-Attestation process. After successful validation and verification of the RA_certificate, the TSTB is challenged with a nonce by the Video-Server. Based upon this nonce a CertifiedKey CK is generated upon an AIK using the TPM. The resulting **CertifyKeyValidation** is presented to the Video-Server which checks the included keyCertification against the challenged nonce and validates and verifies the included AIK-Certificate.

The Video-Server then binds both seeds using CK, builds an AES key based on aesSeed and encrypts the PeerList and the EmbeddedInformation Object using it. The resulting data is then sent to the requesting TSTB, where it is unbound and decrypted for further use.

### 1.3.2 **CONSTRUCTOR** PeerListRetrieval

**Description**
constructs a PeerListRetrieval entity

**Settings**
none

**Parameters**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Vserver_IP</td>
<td>Video-Server's IP address</td>
</tr>
<tr>
<td>Int</td>
<td>Vserver_Port</td>
<td>Video-Server's port</td>
</tr>
<tr>
<td>String</td>
<td>AIKtag</td>
<td>label of the AIK whose RA-Certificate is presented to the Video-Server</td>
</tr>
<tr>
<td>String</td>
<td>AIKpwd</td>
<td>AIK password</td>
</tr>
<tr>
<td>String</td>
<td>SRKpwd</td>
<td>SRK password</td>
</tr>
<tr>
<td>String</td>
<td>CKtag</td>
<td>label to store created CertifiedKey under</td>
</tr>
<tr>
<td>String</td>
<td>CKpwd</td>
<td>password for CertifiedKey</td>
</tr>
<tr>
<td>String</td>
<td>reqVideoName</td>
<td>name of the requested video</td>
</tr>
</tbody>
</table>
1.3.3 PUBLIC run

**Description**  
initiates the protocol by subsequently traversing the protocol states

**Settings**  
none

**Parameters**  
none

**Output**  
1.4 Vclient

1.4.1 Description

Vclient is a wrapper implementing every process needed to actually download and watermark a video. Acting as a demonstrator it is fully automated and accepts no user input. Hence every parameter is inherited from Settings. Nevertheless the user is prompted each time a process is being started, illustrating the parameters passed to the corresponding module.

AIK-Certification

In order to preserve privacy AIKs (Attestation Identity Keys) are being generated using the TPM. These keys allow for pseudonymity and prevent tracking user's download statistics by other entities. They are derived from an already trusted unique key - the endorsement key (EK) of the TPM - and are certified by an issuing PCA (Privacy CA) to become trustworthy to other TSTB's.

This process is fully implemented in References. Please refer to ethemba's documentation for further information on this process.
Remote-Attestation

Trusted Set-Top-Boxes have to prove being unaltered in order to gain access to any data used for video sharing inside the TSTB’s network. This is done by remote attesting a TSTB’s state using IMA (Integrity Measurement Architecture) provided by TPM modules. IMA allows for reliably measuring each program run on a system. The resulting measurement list is revised by an attesting entity which - in case of an unaltered system - issues a so-called RA-Certificate. These certificates have a limited time-frame of validity.

This process is fully implemented in References. Please refer to ethemba’s documentation for further information on this process.

Peer-List-Retrieval

To outsource system load from the servers to the clients, videos are watermarked inside a TSTB. This implies transporting the unmarked videos to the client. Thus it is essential to secure this channel using encryption. To base security on the trustworthy TPM, the encryption key itself is being encrypted for each individual TPM. In addition the seed to base watermarking operations on is being encrypted for each individual TPM too.

This dataset including a PeerList containing peer ⇔ chunks mappings and an EmbeddedInformation object containing a pre-processed message to be embedded into the video, are requested by a TSTB from the Video-Server using the PeerListRetrieval protocol. The requesting TSTB has to attest being unaltered using a RA-Certificate. After successful attestation an individual key is generated inside the TPM (based on a nonce challenged by the Video-Server). This key is used to bind the encryption key and watermark seed to a TSTB’s TPM.

See PeerListRetrieval for further information on this process.

Video-Chunk-Retrieval

This is the actual part of sharing video chunks among Trusted Set-Top-Boxes. TSTBs requesting video chunks have to attest being unaltered using a RA-Certificate. After successful attestation the requested video chunks are being transferred.

See VideoChunkRetrieval for further information on this process.

Decryption/Watermarking

After each video chunk is successfully downloaded (using a download strategy provided by the PeerList), they have to be decrypted and watermarked. This is done by calling PUBLIC decryptDirectory (using the unbound encryption key) and PUBLIC markDirectory (using the unbound watermark seed) provided in MJPEGmovie.

See MJPEGmovie for further information on this process.
1.4.2 **PUBLIC main**

**Description**  
Called when used in command-line.

**Settings**  
PCAdefault_AIKtag, AIKpwd, SRKpwd, OwnerPwd, CKtag, CKpwd, PCAServerIP, PCAServerPort, RAServerIP, RAServerPort, Vserver_IP, Vserver_Port, VCT_Port, Import_defaultVideoName, Import_defaultOutputDir, Import_defaultLeadingZeroes, Import_defaultFramesPerSec

**Parameters**  
none
1.5 VideoChunkRetrieval

VideoChunkRetrieval is the client-side implementation of the VideoChunkRetrieval protocol which is used between two Trusted Set-Top-Boxes to share AES encrypted video chunks. To obtain chunks from another peer, a TSTB has to present a valid RA certificate - which was previously obtained in a Remote-Attestation process - to the target peer.

After successful validation and verification on the server side, all required chunks of a video are consecutively requested by the client and delivered by the server. These chunks are stored in a given directory.

Note: A download strategy containing a sorted list of peer ↔ chunks mappings can be derived.
from a PeerList using the getDownloadStrategy method. Such a strategy is sorted by the number of chunks one peer holds, prioritizing peers with many chunks.

### 1.5.2 CONSTRUCTOR VideoChunkRetrieval

**Description**

constructs a VideoChunkRetrieval entity

**Settings**

none

**Parameters**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRING</td>
<td>Peer_IP target peer's IP address</td>
</tr>
<tr>
<td>INT</td>
<td>Peer_Port target peer's port</td>
</tr>
<tr>
<td>STRING</td>
<td>AIKtag label of the AIK whose RA-Certificate is presented to the target peer</td>
</tr>
<tr>
<td>STRING</td>
<td>reqVideoName name of the requested video</td>
</tr>
<tr>
<td>INT[]</td>
<td>chunkIDs array of chunkIDs to be requested from the target peer</td>
</tr>
<tr>
<td>STRING</td>
<td>videoDir directory to store videoChunks in</td>
</tr>
</tbody>
</table>

### 1.5.3 PUBLIC run

**Description**

initiates the protocol by subsequently traversing the protocol states

**Settings**

none

**Parameters**

none

**Output**

BOOLEAN[] containing download result for each requested chunkID
1.6 VideoChunkTransmission

VideoChunkTransmission is a server-side representation of the VideoChunkRetrieval protocol. It is separated into different methods representing each state of the protocol. The methods are called subsequently.

→ see VideoChunkRetrieval for a detailed description of the VideoChunkRetrieval protocol
1.7 Vserver

The server is separated into different methods representing each state of the PeerListRetrieval protocol. The methods are called subsequently.

→ see PeerListRetrieval for a detailed description of the PeerListRetrieval protocol
2 modules

DE.FRAUNHOFER.SIT.TC.TSTB.MODELES

This package contains modules providing modular watermarking methods.

FridrichPatternGenerator generates FridrichPatterns based on an initial seed. These patterns are applied to JPEG blocks to watermark a JPEG image.

Accordingly JPEGmarker allows for convenient watermarking of a JPEG image providing it with just a seed to base the whole watermarking process on.
2.1 FridrichPatternGenerator

2.1.1 Description

FridrichPatternGenerator provides access to Fridrich patterns that are either derived from a given initial-seed or from a given initial-pattern.

The generation process involves 4 steps:

1. **randomize** or **copy**: generate pseudo-random pattern from initial-seed or copy initial-pattern (depends on CONSTRUCTOR usage)

2. **filter**: transform pattern from high-frequent to low-frequent (applied 3x)

3. **strengthen**: strengthen pattern by increasing “color” depth

4. **smooth**: smooth pattern by averaging

2.1.2 **CONSTRUCTOR** FridrichPatternGenerator

**Description**

creates a Fridrich-Pattern of dimension ‘dim’ using seed ‘K’

**Settings**

none

**Parameters**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>dim</td>
<td>dimension of the generated Fridrich pattern</td>
</tr>
<tr>
<td>BYTE[]</td>
<td>K</td>
<td>seed to base pseudo-random initial-pattern on</td>
</tr>
</tbody>
</table>

Figure 2.1: constructing a FridrichPattern
2.1.3 **CONSTRUCTOR** FridrichPatternGenerator

**Description**
creates a Fridrich-Pattern of dimension 'dim' derived from 'initPattern' (no pseudo-random initPattern is being generated)

**Settings**
none

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT dim</td>
<td>dimension of the generated Fridrich pattern</td>
</tr>
<tr>
<td>INT[][] initPattern</td>
<td>initial-pattern to use</td>
</tr>
</tbody>
</table>

2.1.4 **PUBLIC** get

**Description**
returns the generated Fridrich pattern as INT[][]

**Settings**
none

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>None</th>
</tr>
</thead>
</table>

**Output**

INT[][] generated Fridrich pattern
2.2 JPEGmarker

2.2.1 Description

JPEGmarker provides easy access to watermarking JPEG images. It can watermark already loaded JPEGIMAGEs or JPEG files on the disk and handles CRC-checking methods. It therefore makes use of JPEGimage's watermarking capabilities. Based on an initial seed K, two further seeds (K0 and K1) are being pseudo-randomly generated. These seeds are used to create two FridrichPatterns (see FridrichPatternGenerator). One Pattern for embedding a 0-Bit and the other one for embedding a 1-Bit. The watermark message for a frame is being embedded JPEG blockwise bit for bit.

Note: Before applying a payload bitstring to an image, a CRC-8 checksum is calculated over that payload. The concatenation of payload and CRC-8 checksum is then embedded into the image.

Note: When reading out embedded bits, the CRC-8 checksum is used to verify the payload's integrity.

Note: JPEGimage's embedWatermark method makes use of Smooth-Edge-Detection in order to determine JPEG blocks, that are too "smooth" to imperceptibly embed a bit into. These blocks are skipped to improve imperceptibility of the watermark.
### 2.2.2 **CONSTRUCTOR** JPEGmarker

**Description**
constructs JPEGmarker from a given file

**Settings**
none

**Parameters**
- **STRING** pathToImage: path to image to be watermarked or read out
- **BYTE[]** K: seed to base watermarking process on

### 2.2.3 **CONSTRUCTOR** JPEGmarker

**Description**
constructs JPEGmarker directly from JPEGimage

**Settings**
none

**Parameters**
- **JPEGIMAGE** jpeg: image to be watermarked or read out
- **BYTE[]** K: seed to base watermarking process on

### 2.2.4 **PUBLIC** embedBits

**Description**
eembeds given (boolean[])bitstring into the image

**Settings**
none

**Parameters**
- **BOOLEAN[]** bits: (payload-)bits to be embedded in this image

### 2.2.5 **PUBLIC** readBits

**Description**
reads embedded bits from an image

**Settings**
none

**Parameters**
- **INT** bitLength: number of (payload-)bits embedded in this image
Output
BOOLEAN[] read bitstring or NULL if CRC-check did NOT succeed

2.2.6 **PUBLIC** saveJPEG

**Description**
saves JPEG to the disk

**Settings**
none

**Parameters**
none

2.2.7 **PUBLIC** saveJPEG

**Description**
saves JPEG under another filename

**Settings**
none

**Parameters**

| STRING fileName | filename to store JPEG under |
This package contains networking classes for sending and receiving data via TCP/IP.
3.1 NetCommand

3.1.1 Description

NetCommand provides a mapping between symbolic names and command codes (INT) for convenient use of NETENTITY objects. It extends NETCOMMANDs provided by ethemba and its network-module NETENTITY (see References).

- PLR_REQUEST
- VCR_REQUEST
This package contains data types for access to mathematical operations on JPEG-Blocks. *EmbeddedInformation* pre-processes a watermark-message to be directly embedded into a video by splitting, sequencing and randomizing it over all frames of the video. *PeerList* furthermore holds a mapping of sharing peers for every chunk of a video.
4.1 BitString

4.1.1 Description

4.1.2 PUBLIC getBits

Description
converts a STRING bitstring into a BOOLEAN[] bitstring

Settings
none

Parameters

<table>
<thead>
<tr>
<th>STRING bitstring</th>
<th>STRING bitstring to be converted into a BOOLEAN[] bitstring</th>
</tr>
</thead>
</table>

Output
BOOLEAN[] converted bitstring

4.1.3 PUBLIC getBitString

Description
converts a BOOLEAN[] bitstring into a STRING bitstring

Settings
none

Parameters

<table>
<thead>
<tr>
<th>BOOLEAN[] bits</th>
<th>BOOLEAN[] bitstring to be converted into a STRING bitstring</th>
</tr>
</thead>
</table>

Output
STRING converted bitstring

4.1.4 PUBLIC getBitString

Description
converts a BOOLEAN[] bitstring into a STRING bitstring applying the given pattern

Settings
none

Parameters

<table>
<thead>
<tr>
<th>BOOLEAN[] bits</th>
<th>BOOLEAN[] bitstring to be converted into a STRING bitstring</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>STRING pattern</th>
<th>“.” specifies the actual bits, other chars may be used for layouting</th>
</tr>
</thead>
</table>
Output
STRING converted bitstring

4.1.5 PUBLIC concatBits

Description
concatenates two BOOLEAN[] bitstrings

Settings
none

Parameters
| BOOLEAN[] a | head |
| BOOLEAN[] b | tail |

Output
BOOLEAN[] concatenated BOOLEAN[] bitstring: a|b

4.1.6 PUBLIC concatBits

Description
concatenates a BOOLEAN[] bitstring with a STRING bitstring

Settings
none

Parameters
| BOOLEAN[] a | head |
| STRING b   | tail |

Output
BOOLEAN[] concatenated BOOLEAN[] bitstring: a|b

4.1.7 PUBLIC firstBits

Description
returns the first 'count' bits of the given BOOLEAN[] bitstring

Settings
none

Parameters
| BOOLEAN[] a | bitstring |
| INT count   | number of bits to return |
Output
BOOLEAN[] first 'count' bits of the given bitstring

4.1.8 PUBLIC firstBits

Description
returns a snippet of bits of the given BOOLEAN[] bitstring

Settings
none

Parameters
| BOOLEAN[] a | bitstring |
| INT start | position to start snipping |
| INT count | number of bits to snip |

Output
BOOLEAN[] snippet of bits of the given bitstring

4.1.9 PUBLIC lastBits

Description
returns the last 'count' bits of the given BOOLEAN[] bitstring

Settings
none

Parameters
| BOOLEAN[] a | bitstring |
| INT count | number of bits to return |

Output
BOOLEAN[] last 'count' bits of the given bitstring

4.1.10 PUBLIC createCRC4

Description
calculates CRC-4 checksum for given BOOLEAN[] bitstring

Settings
none

Parameters
| BOOLEAN input | bitstring to calculate CRC-4 checksum of |

Output
BOOLEAN CRC-4 checksum of the given bitstring
4.1.11 PUBLIC verifyCRC4

**Description**
verifies a given BOOLEAN[] bitstring against a given CRC-4

**Settings**
none

**Parameters**

| BOOLEAN[] | input     | bitstring to verify |
| BOOLEAN[] | crc       | CRC-4 checksum to check against |

**Output**
returns TRUE if CRC-4(input) equals crc

4.1.12 PUBLIC createCRC8

**Description**
calculates CRC-8 checksum for given BOOLEAN[] bitstring

**Settings**
none

**Parameters**

| BOOLEAN | input | bitstring to calculate CRC-8 checksum of |

**Output**
BOOLEAN CRC-8 checksum of the given bitstring

4.1.13 PUBLIC verifyCRC8

**Description**
verifies a given BOOLEAN[] bitstring against a given CRC-8

**Settings**
none

**Parameters**

| BOOLEAN[] | input     | bitstring to verify |
| BOOLEAN[] | crc       | CRC-8 checksum to check against |

**Output**
returns TRUE if CRC-8(input) equals crc
4.2 Block

4.2.1 Description

Block is a two-dimensional matrix of dimension \texttt{Settings}.dim. It represents JPEG-Blocks as well as FridrichPatterns (see \texttt{FridrichPatternGenerator}) and allows for applying mathematical operations like addition, subtraction, multiplication and also cross-correlation with other blocks. The data-type (integer or floating point) is kept abstract.

There are two sub-classes implementing type-specific versions of \texttt{Block}:

- \texttt{IntBlock}
- \texttt{FloatBlock}

4.2.2 \texttt{CONSTRUCTOR} Block

\textbf{Description}

constructs a block with initial dimension \texttt{Settings}.dim

\textbf{Settings}

\texttt{dim}

\textbf{Parameters}

\texttt{none}

4.2.3 \texttt{PUBLIC} get

\textbf{Description}

returns value of the requested position

\textbf{Settings}

\texttt{none}

\textbf{Parameters}

\begin{tabular}{ll}
  \texttt{INT} & \texttt{i} & row \\
  \texttt{INT} & \texttt{k} & column \\
\end{tabular}

\textbf{Output}

\texttt{OBJECT} returns value of the requested position
4.2.4 **PUBLIC getI**

**Description**
returns integer value of the requested position

**Settings**
none

**Parameters**

<table>
<thead>
<tr>
<th>INT i</th>
<th>row</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT k</td>
<td>column</td>
</tr>
</tbody>
</table>

**Output**
INT returns integer value of the requested position

4.2.5 **PUBLIC getF**

**Description**
returns float value of the requested position

**Settings**
none

**Parameters**

<table>
<thead>
<tr>
<th>INT i</th>
<th>row</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT k</td>
<td>column</td>
</tr>
</tbody>
</table>

**Output**
FLOAT returns float value of the requested position

4.2.6 **PUBLIC set**

**Description**
sets value of the given position to the given one

**Settings**
none

**Parameters**

<table>
<thead>
<tr>
<th>INT i</th>
<th>row</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT k</td>
<td>column</td>
</tr>
</tbody>
</table>

| OBJECT value | value to set |
4.2.7 **PUBLIC** setl

**Description**
sets value of the given position to the given one (integer)

**Settings**
none

**Parameters**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>i</td>
<td>row</td>
</tr>
<tr>
<td>INT</td>
<td>k</td>
<td>column</td>
</tr>
<tr>
<td>INT</td>
<td>value</td>
<td>value to set</td>
</tr>
</tbody>
</table>

4.2.8 **PUBLIC** setF

**Description**
sets value of the given position to the given one (float)

**Settings**
none

**Parameters**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>i</td>
<td>row</td>
</tr>
<tr>
<td>INT</td>
<td>k</td>
<td>column</td>
</tr>
<tr>
<td>FLOAT</td>
<td>value</td>
<td>value to set</td>
</tr>
</tbody>
</table>

4.2.9 **PUBLIC** multiply

**Description**
multiplies the whole block with a given value

**Settings**
none

**Parameters**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJECT</td>
<td>value</td>
<td>multiplier</td>
</tr>
</tbody>
</table>

4.2.10 **PUBLIC** add

**Description**
adds a given value to a given position (i, k)

**Settings**
none
4.2 Block

### Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>i</td>
<td>row</td>
</tr>
<tr>
<td>INT</td>
<td>k</td>
<td>column</td>
</tr>
<tr>
<td>OBJECT</td>
<td>value</td>
<td>value to add</td>
</tr>
</tbody>
</table>

#### 4.2.11 PUBLIC sub

**Description**

Subtracts a given value from a given position \((i, k)\)

**Settings**

none

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>i</td>
<td>row</td>
</tr>
<tr>
<td>INT</td>
<td>k</td>
<td>column</td>
</tr>
<tr>
<td>OBJECT</td>
<td>value</td>
<td>value to substract</td>
</tr>
</tbody>
</table>

#### 4.2.12 PUBLIC addBlock

**Description**

Adds a given block

**Settings**

none

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCK</td>
<td>addBlock</td>
<td>block to add</td>
</tr>
</tbody>
</table>

#### 4.2.13 PUBLIC addBlock

**Description**

Adds a given block starting at a given index (ZigZag order)

**Settings**

none

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCK</td>
<td>addBlock</td>
<td>block to add</td>
</tr>
<tr>
<td>INT</td>
<td>start</td>
<td>index to start adding from (ZigZag order)</td>
</tr>
</tbody>
</table>
4.2.14 **PUBLIC** subBlock

**Description**
subtracts a given block

**Settings**
none

**Parameters**
<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCK</td>
<td>subBlock</td>
<td>block to substract</td>
</tr>
</tbody>
</table>

4.2.15 **PUBLIC** subBlock

**Description**
subtracts a given block starting at a given index (ZigZag order)

**Settings**
none

**Parameters**
<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCK</td>
<td>subBlock</td>
<td>block to add</td>
</tr>
<tr>
<td>INT</td>
<td>start</td>
<td>index to start substracting from (ZigZag order)</td>
</tr>
</tbody>
</table>

4.2.16 **PUBLIC** doCrossCorrelation

**Description**
calculates cross-correlation with given Block

\[
Cor(b, p) = \frac{\sum_{i=f}^{g} b_i \cdot p_i}{\sqrt{\sum_{i=f}^{g} b_i^2 \cdot \sum_{i=f}^{g} p_i^2}}
\]

\[
f = \text{index to start from (ZigZag order)}
\]
\[
g = \text{Settings}.\text{dim}^2 - 1
\]

**Settings**
none

**Parameters**
<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCK</td>
<td>p</td>
<td>Block to calculate cross-correlation with</td>
</tr>
<tr>
<td>INT</td>
<td>start</td>
<td>index to start from</td>
</tr>
</tbody>
</table>

**Output**
FLOAT cross-correlation value
4.2.17 **PUBLIC equals**

**Description**
checks for equality against a given block

**Settings**
none

**Parameters**

<table>
<thead>
<tr>
<th>BLOCK</th>
<th>block2</th>
<th>block to check for equality</th>
</tr>
</thead>
</table>

**Output**
TRUE if blocks match, FALSE otherwise
4.3 BlockVector

4.3.1 Description

BlockVector is a one-dimensional representation of a Block. It provides simplified implementations of mathematical functions that access a Block in ZigZagOrder (e.g. Cross-Correlation).

\[
\begin{pmatrix}
a & b & c \\
d & e & f \\
g & h & i \\
\end{pmatrix}
\xrightarrow{\text{ZigZag order}}
\begin{pmatrix}
a & b & d & g & e & c & f & h & i \\
\end{pmatrix}
\]

Like Block it is kept abstract concerning the data-type (integer or floating point).

There are two sub-classes implementing type-specific versions of BlockVector:

- `IntBlockVector`
- `FloatBlockVector`

4.3.2 **CONSTRUCTOR** BlockVector

Description constructs a BlockVector with initial dimension `Settings.dim`

Settings

dim

Parameters

| none |

4.3.3 **PUBLIC** get

Description returns value of the requested position

Settings

none

Parameters

| INT i | index |

Output

`OBJECT` returns value of the requested position
4.3.4 **PUBLIC** getI

**Description**
returns integer value of the requested position

**Settings**
one

**Parameters**

<table>
<thead>
<tr>
<th>INT</th>
<th>index</th>
</tr>
</thead>
</table>

**Output**

INT  returns integer value of the requested position

4.3.5 **PUBLIC** getF

**Description**
returns float value of the requested position

**Settings**
one

**Parameters**

<table>
<thead>
<tr>
<th>INT</th>
<th>index</th>
</tr>
</thead>
</table>

**Output**

FLOAT  returns float value of the requested position

4.3.6 **PUBLIC** set

**Description**
sets value of the given position to the given one

**Settings**
one

**Parameters**

<table>
<thead>
<tr>
<th>INT</th>
<th>index</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>OBJECT</th>
<th>value</th>
</tr>
</thead>
</table>

| value to set |

4.3.7 **PUBLIC** setI

**Description**
sets value of the given position to the given one (integer)
### 4.3.8 PUBLIC setF

**Description**

sets value of the given position to the given one (float)

**Settings**

none

**Parameters**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>index</td>
</tr>
<tr>
<td>INT</td>
<td>value</td>
</tr>
</tbody>
</table>

### 4.3.9 PUBLIC doCrossCorrelation

**Description**

calculates cross-correlation with given BlockVector

\[
Cor(b, p) = \frac{\sum_{i=f}^{g} b_i \cdot p_i}{\sqrt{\sum_{i=f}^{g} b_i^2 \cdot \sum_{i=f}^{g} p_i^2}}
\]

- \( f \) = index to start from (ZigZag order)
- \( g = Settings\text{.dim}^2 - 1 \)

**Settings**

none

**Parameters**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCKVECTOR</td>
<td>BlockVector to calculate cross-correlation with</td>
</tr>
<tr>
<td>INT</td>
<td>index to start from</td>
</tr>
</tbody>
</table>

**Output**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOAT</td>
<td>cross-correlation value</td>
</tr>
</tbody>
</table>
4.4 EmbeddedInformation

4.4.1 Description

EmbeddedInformation pre-processes a watermark message to be embedded into a video. The message therefore is split into chunks of equal size (defined in Settings.payloadBitsPerFrame) which are then randomly distributed over the whole video. To allow for successful reassembly of the message when reading out the watermark, the message slices are numbered before being shuffled. The message slices (sequence number | payload) are then applied to their assigned frame.

Note: The number of sequence number bits is defined in Settings.sequenceBitsPerFrame.

4.4.2 CONSTRUCTOR EmbeddedInformation

Description
constructs an information to be embedded as a watermark over a whole video

Settings
sequenceBitsPerFrame, payloadBitsPerFrame

Parameters
| BOOLEAN[] message | message to be embedded |
| INT numFrames | number of frames in the movie |
4.4.3 **PUBLIC** getMessage

**Description**  
returns message to be embedded

**Settings**  
none

**Parameters**  
none

**Output**  
BOOLEAN  
message to be embedded

4.4.4 **PUBLIC** getInformation

**Description**  
returns whole information (message spread over all frames)

**Settings**  
none

**Parameters**  
none

**Output**  
BOOLEAN[][]  
whole information (message spread over all frames)

4.4.5 **PUBLIC** getInformation

**Description**  
returns the information that is to be embedded into the given frame

**Settings**  
none

**Parameters**  
INT  
frame  
frame whose information is to be returned

**Output**  
BOOLEAN[]  
information that is to be embedded into the given frame
4.4.6 PUBLIC getFrameNum

Description
returns the number of frames

Settings
none

Parameters
none

Output
INT number of frames
4.5 FloatBlock

FloatBlock is a floating point implementation of Block.

4.5.1 CONSTRUCTOR FloatBlock

Description
constructs a block with initial dimension Settings.dim

Settings
dim

Parameters

none

4.5.2 CONSTRUCTOR FloatBlock

Description
constructs a block from a given initial Block

Settings
dim

Parameters

FLOAT[[]] initBlock initial block to construct FloatBlock from

4.5.3 CONSTRUCTOR FloatBlock

Description
constructs a block from a given initial Block

Settings
dim

Parameters

INT[[]] initBlock initial block to construct FloatBlock from
4.5.4 **CONSTRUCTOR** FloatBlock

**Description**
constructs a block from a given BlockVector

**Settings**
dim

**Parameters**
```
FLOATBLOCKVECTOR initVec - initial BlockVector to construct FloatBlock from
```

4.5.5 **PUBLIC** doDCT

**Description**
performs Discrete Cosine Transform (DCT)

\[
F_{x,y} = \frac{2 \cdot C(x) \cdot C(y)}{N} \cdot \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f_{i,j} \cdot \cos \left( \frac{(2i + 1) \cdot x \cdot \pi}{2 \cdot N} \right) \cdot \cos \left( \frac{(2j + 1) \cdot y \cdot \pi}{2 \cdot N} \right)
\]

\[
C(n) = \begin{cases} 
\frac{1}{\sqrt{2}}, & n = 0 \\
1, & n \neq 0 
\end{cases}
\]

**Settings**
dctCoefficients, dim

**Parameters**

**Output**
FLOATBLOCK DCT transformed block

4.5.6 **PUBLIC** doIDCT

**Description**
performs Inverse Discrete Cosine Transform (IDCT)

\[
f_{i,j} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \cdot \frac{2 \cdot C(x) \cdot C(y)}{N} \cdot F_{x,y} \cdot \cos \left( \frac{(2i + 1) \cdot x \cdot \pi}{2 \cdot N} \right) \cdot \cos \left( \frac{(2j + 1) \cdot y \cdot \pi}{2 \cdot N} \right)
\]

\[
C(n) = \begin{cases} 
\frac{1}{\sqrt{2}}, & n = 0 \\
1, & n \neq 0 
\end{cases}
\]
Settings
dctCoefficients, dim

Parameters
none

Output
INTBLOCK IDCT transformed block

4.5.7 PUBLIC tolIntBlock

Description
converts a FloatBlock into an IntBlock

Settings
dim

Parameters
none

Output
INTBLOCK converted IntBlock
4.6 FloatBlockVector

4.6.1 Description

FloatBlockVector is a floating point implementation of BlockVector.

4.6.2 Constructor FloatBlockVector

Description

constructs a BlockVector with initial dimension Settings.dim

Settings

dim

Parameters

none

4.6.3 Constructor FloatBlockVector

Description

constructs a BlockVector from a given initial BlockVector

Settings

dim

Parameters

INT[] initVector initial BlockVector to construct FloatBlockVector from

4.6.4 Constructor FloatBlockVector

Description

constructs a BlockVector from a given initial Block

Settings

zigZagOrder

Parameters

BLOCK initBlock initial Block to construct FloatBlockVector from
4.7 IntBlock

4.7.1 Description

IntBlock is an integer implementation of Block.

4.7.2 CONSTRUCTOR IntBlock

Description
constructs a block with initial dimension Settings.dim

Settings
dim

Parameters
none

4.7.3 CONSTRUCTOR IntBlock

Description
constructs a block from a given initial Block

Settings
dim

Parameters

<table>
<thead>
<tr>
<th>INT[]</th>
<th>initBlock</th>
<th>initial block to construct IntBlock from</th>
</tr>
</thead>
</table>

4.7.4 CONSTRUCTOR IntBlock

Description
constructs a block from a given BlockVector

Settings
dim

Parameters

<table>
<thead>
<tr>
<th>INTBLOCKVECTOR</th>
<th>initVec-</th>
<th>initial BlockVector to construct IntBlock from</th>
</tr>
</thead>
</table>
4.7.5 **PUBLIC** doDCT

**Description**
performs Discrete Cosine Transform (DCT)

\[
F_{x,y} = \frac{2 \cdot C(x) \cdot C(y)}{N} \cdot \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f_{i,j} \cdot \cos\left(\frac{(2i + 1) \cdot x \cdot \pi}{2 \cdot N}\right) \cdot \cos\left(\frac{(2j + 1) \cdot y \cdot \pi}{2 \cdot N}\right)
\]

\[
C(n) = \begin{cases} 
\frac{1}{\sqrt{2}}, & n = 0 \\
1, & n \neq 0 
\end{cases}
\]

**Settings**
dctCoefficients, dim

**Parameters**
none

**Output**
FLOATBLOCK DCT transformed block
4.8 IntBlockVector

4.8.1 Description

IntBlockVector is an integer implementation of BlockVector.

4.8.2 Constructor IntBlockVector

Description
constructs a BlockVector with initial dimension Settings.dim

Settings
dim

Parameters

4.8.3 Constructor IntBlockVector

Description
constructs a BlockVector from a given initial BlockVector

Settings
dim

Parameters

4.8.4 Constructor IntBlockVector

Description
constructs a BlockVector from a given initial Block

Settings
zigZagOrder

Parameters

4.9 JPEGImage

4.9.1 Description

JPEGImage provides methods for applying watermarks to and reading out watermarks from JPEG images. It uses Smooth-Edge-Detection to filter JPEG blocks that can not be watermarked without negatively affecting watermark's perceptibility.

4.9.2 Constructor JPEGImage

Description

constructs JPEGImage from a jpeg-file

Settings

dim

Parameters

| String fileName | jpeg-file to be loaded |

4.9.3 Constructor JPEGImage

Description

constructs JPEGImage directly from a BufferedImage

Settings

dim

Parameters

| BufferedImage image | BufferedImage to be used |

4.9.4 Private doSmoothEdgeDetection

Description

doSmoothEdgeDetection calculates the so-called Smooth-Edge-Detection value for a given FloatBlock. It is used to determine if a block is capable of embedding a watermark without negatively affecting watermark's perceptibility. “Smooth” blocks (low-frequent) are less capable of embedding imperceptible watermarks while “edgy” blocks (high-frequent) provide enough potential to unnoticeably embed a watermark-bit in.

Settings

SED_smoothScale, SED_edgeScale, SED_offset

Parameters

| FloatBlock fb | FloatBlock to calculate SED-value for |
**Output**
FLOAT Smooth-Edge-Detection value estimating how “smooth” or “edgy” a block is

**4.9.5 PUBLIC embedWatermark**

**Description**
embeds a given bitstring into the image using FridrichPatterns

**Settings**
dim, SED_edginess, SED_showResult, startCoefficient

**Parameters**
- BOOLEAN[] bits: bits to be embedded into the image
- FLOAT BLOCK pattern0: FridrichPattern to apply for embedded 0’s
- FLOAT BLOCK pattern1: FridrichPattern to apply for embedded 1’s

**4.9.6 PUBLIC readWatermark**

**Description**
reads the embedded bitstring from an image using FridrichPatterns

**Settings**
SED_edginess, startCoefficient

**Parameters**
- INT bitlength: number of bits embedded
- FLOAT BLOCK pattern0: FridrichPattern to apply for embedded 0’s
- FLOAT BLOCK pattern1: FridrichPattern to apply for embedded 1’s

**Output**
BOOLEAN[] read bitstring

**4.9.7 PUBLIC save**

**Description**
saves JPEGimage

**Settings**
none

**Parameters**
none
4.9.8 PUBLIC save

**Description**

saves JPEGimage under another filename

**Settings**

none

**Parameters**

| STRING fileName | filename to save JPEGimage under |
4.10 MJPEGmovie

4.10.1 Description

MJPEGmovie provides methods for applying watermarks to and reading out watermarks from whole MJPEG videos. It also allows for extracting the frames of a MJPEG video to JPEG images. These frames can further be encrypted into chunks holding several frames (defined in Settings.framesPerChunk, which can be decrypted in a separate process.

Note: To separate frames from chunks, each type has its own filename-prefix (defined in Settings.frameNamePrefix and Settings.chunkNamePrefix).

4.10.2 CONSTRUCTOR MJPEGmovie

Description
constructs MJPEGmovie from a mjpeg-file using JMF (Java Media Framework)

Settings
none

Parameters

| STRING  fileName | mjpeg-file to be loaded |

4.10.3 PUBLIC getFrame

Description
returns requested frame as JPEGimage

Settings
none

Parameters

| INT  frame | frame number to be returned |

Output
JPEGimage requested frame as JPEGimage

4.10.4 PUBLIC numFrames

Description
returns number of frames in the mjpeg-video

Settings
none
4.10.5 **PUBLIC close**

**Description**
closes the video

**Settings**
none

**Parameters**
none

4.10.6 **PUBLIC readWatermark**

**Description**
reads out the embedded watermark

**Settings**
sequenceBitsPerFrame, payloadBitsPerFrame, Import_defaultLeadingZeroes

**Parameters**

<table>
<thead>
<tr>
<th>TYPE</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYTE[]</td>
<td>k</td>
<td>watermark-seed to be used</td>
</tr>
</tbody>
</table>

**Output**
STRING read out watermark

4.10.7 **PUBLIC convertJPEGtoMJPEG**

**Description**
converts a directory containing extracted JPEGs to one MJPEG video

**Settings**
frameNamePrefix

**Parameters**

<table>
<thead>
<tr>
<th>TYPE</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRING</td>
<td>inputDir</td>
<td>directory containing extracted JPEGs</td>
</tr>
<tr>
<td>INT</td>
<td>frameRate</td>
<td>frame-rate to be used for the resulting video</td>
</tr>
<tr>
<td>STRING</td>
<td>outputFile</td>
<td>output-filename of the resulting video</td>
</tr>
</tbody>
</table>
4.10.8 **PUBLIC** `convertMJPEGtoJPEG`

**Description**
extracts frames of a given MJPEG video as JPEGs

**Settings**
frameNamePrefix, Import_defaultLeadingZeroes

**Parameters**
- `BYTE[][] inputVideo`: MJPEG video to be converted
- `STRING outputDir`: location for storing extracted JPEGs

**Output**
`STRING[]`: list of extracted JPEGs

4.10.9 **PUBLIC** `markDirectory`

**Description**
embeds a watermark into MJPEG-frames (.jpg)

**Settings**
frameNamePrefix

**Parameters**
- `STRING inputDir`: directory containing MJPEG-frames in JPEG format
- `STRING outputDir`: outputDir directory to store watermarked frames in
- `BYTE[] k`: watermark-seed to be used
- `EMBEDDEDINFORMATION embeddedInformation`: whole watermark information to be embedded

4.10.10 **PUBLIC** `encryptDirectory`

**Description**
encrypts a directory containing MJPEG-Frames (.jpg) into AES-Block-Containers (.jpg.aes)

**Settings**
frameNamePrefix, framesPerChunk, chunkNamePrefix, Import_defaultLeadingZeroes

**Parameters**
- `BYTE[][] inputDir`: directory containing MJPEG-frames in JPEG format
- `STRING outputDir`: directory to store encrypted AES-Block-Containers in
- `BYTE[] seed`: AES-seed to be used

**Output**
`STRING[]`: list of encrypted files
4.10.11 **PUBLIC decryptDirectory**

**Description**
decrypts a directory containing AES-Block-Containers (.jpg.aes) into MJPEG-Frames (.jpg)

**Settings**
frameNamePrefix, framesPerChunk, chunkNamePrefix, Import_defaultLeadingZeroes

**Parameters**
<table>
<thead>
<tr>
<th>Type</th>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYTE[]</td>
<td>inputDir</td>
<td>directory containing AES-Block-Containers</td>
</tr>
<tr>
<td>STRING</td>
<td>outputDir</td>
<td>directory to store decrypted MJPEG-frames in</td>
</tr>
<tr>
<td>BYTE[]</td>
<td>seed</td>
<td>AES-seed to be used</td>
</tr>
</tbody>
</table>

**Output**
STRING[] list of decrypted files
4.11 PeerList

4.11.1 Description

PeerList is a mapping table to store peer ⇔ video-chunk mappings. It holds a list of all peers sharing a video, thus allowing a peer to request video-chunks from other peers being in possession of that chunk.

Note: PeerLists are distributed to each Vclient by the Vserver after successful Remote Attestation. Vclients thereupon share video-chunks among themselves using P2P technology, thus reducing server's load.

4.11.2 Constructor PeerList

Description
initializes the PeerList

Settings
none

Parameters

| STRING  | videoName | name of the video this PeerList corresponds to |
| INT     | numChunks | number of chunks for this video |

4.11.3 PUBLIC put

Description
maps the specified peer to the specified chunk

Settings
none

Parameters

| INT     | chunkID | video-chunk's ID |
| STRING  | peer    | peer's IP-Address |

4.11.4 PUBLIC get

Description
returns the peers to which the specified chunk is mapped

Settings
none

Parameters

| INT     | chunkID | a chunkID in the PeerList |
Output
STRING[] the peers to which the specified chunkID is mapped; NULL if the specified chunkID is not mapped to any peer

4.11.5 PUBLIC get

Description
returns the chunks to which the specified peer is mapped

Settings
none

Parameters
| STRING peer | a peer in the PeerList |

Output
INT[] the chunks to which the specified peer is mapped; NULL if the specified peer is not mapped to any chunk

<table>
<thead>
<tr>
<th>Peer</th>
<th>Chunks</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.2.2.19</td>
<td>1, 3, 4, 5, 11, 14, 27, 29</td>
</tr>
<tr>
<td>10.2.2.15</td>
<td>1, 2, 3, 4, 15, 21</td>
</tr>
<tr>
<td>10.2.2.17</td>
<td>3, 7, 9, 20</td>
</tr>
</tbody>
</table>

Table 4.1: example downloadStrategy

4.11.6 PUBLIC getDownloadStrategy

Description
evaluates a download strategy for this video by prioritizing peers holding many chunks (see table 4.1)

Settings
none

Parameters
none

Output
OBJECT[] containing peer <-> chunkIDs mapping (sorted by number of chunks per peer in descending order)
4.11.7 PUBLIC getVideoName

**Description**
returns the name of the video this PeerList corresponds to

**Settings**
none

**Parameters**
none

**Output**
STRING name of the video this PeerList corresponds to

4.11.8 PUBLIC getNumChunks

**Description**
returns the number of chunks of this video

**Settings**
none

**Parameters**
none

**Output**
STRING name of the video this PeerList corresponds to
5 utils

DE.FRAUNHOFER.SIT.TC.TSTB.UTILS

This package contains classes and methods for data conversion and standardized algorithms, that may be accessed in a convenient way.
5.1 Colors

5.1.1 Description

Colors provides methods for color conversion (RGB ⇔ YCbCr ⇔ Lab).

Note: Converting colors from one color space to another produces (small) rounding errors due to 8-bit digital precision. This deviation may accumulate and actually lead to color mismatches between color spaces.

5.1.2 PUBLIC rgb2ycc

Description
convert RGB to YCbCr

Settings
none

Parameters
| INT[] | rgb | RGB color triple |

Output
INT[] converted YCbCr color triple

5.1.3 PUBLIC ycc2rgb

Description
convert YCbCr to RGB

Settings
none

Parameters
| INT[] | ycc | YCbCr color triple |

Output
INT[] converted RGB color triple

5.1.4 PUBLIC rgb2y

Description
convert RGB to luminance value

Settings
none
5.1 Colors

**Parameters**

| INT[] rgb | RGB color triple |

**Output**

INT converted luminance value
5.2 Images

5.2.1 Description
Images provides methods for image-handling, -manipulation and -analysis.

5.2.2 PUBLIC histogramAbs

Description
derives histogram containing absolute values (pixel-count) from a given BufferedImage

Settings
none

Parameters

| BufferedImage | bim | image to derive histogram from |

Output

```
INT[] histogram (absolute values)
```

5.2.3 PUBLIC histogramRel

Description
derives histogram containing relative values (percentage) from a given BufferedImage

Settings
none

Parameters

| BufferedImage | bim | image to derive histogram from |

Output

```
INT[] histogram (relative values)
```

5.2.4 PUBLIC histogramImage

Description
derives histogram as an image from a given BufferedImage

Settings
none

Parameters

| BufferedImage | bim | image to derive histogram from |
Output
BUFFEREDIMAGE histogram image (256*100px)

5.2.5 PUBLIC getPixelY

Description
returns luminance value of a given position in a given BUFFEREDIMAGE

Settings
none

Parameters
<table>
<thead>
<tr>
<th>BUFFEREDIMAGE bim</th>
<th>image</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT x</td>
<td>x-position</td>
</tr>
<tr>
<td>INT y</td>
<td>y-position</td>
</tr>
</tbody>
</table>

Output
INT luminance value

5.2.6 PUBLIC getPixelYCC

Description
returns YCbCr triple of a given position in a given BUFFEREDIMAGE

Settings
none

Parameters
<table>
<thead>
<tr>
<th>BUFFEREDIMAGE bim</th>
<th>image</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT x</td>
<td>x-position</td>
</tr>
<tr>
<td>INT y</td>
<td>y-position</td>
</tr>
</tbody>
</table>

Output
INT[] YCbCr triple

5.2.7 PUBLIC getPixelRGB

Description
returns RGB triple of a given position in a given BUFFEREDIMAGE

Settings
none

Parameters
<table>
<thead>
<tr>
<th>BUFFEREDIMAGE bim</th>
<th>image</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT x</td>
<td>x-position</td>
</tr>
<tr>
<td>INT y</td>
<td>y-position</td>
</tr>
</tbody>
</table>
Output
INT[] RGB triple

5.2.8 PUBLIC setPixelRGB

Description
sets RGB triple (integer) on a position in a given BUFFEREDIMAGE

Settings
none

Parameters
<table>
<thead>
<tr>
<th>BUFFEREDIMAGE</th>
<th>bim</th>
<th>image</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>x</td>
<td>x-position</td>
</tr>
<tr>
<td>INT</td>
<td>y</td>
<td>y-position</td>
</tr>
<tr>
<td>INT[]</td>
<td>rgb</td>
<td>RGB triple</td>
</tr>
</tbody>
</table>

5.2.9 PUBLIC setPixelRGB

Description
sets RGB triple (floating point) on a position in a given BUFFEREDIMAGE

Settings
none

Parameters
<table>
<thead>
<tr>
<th>BUFFEREDIMAGE</th>
<th>bim</th>
<th>image</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>x</td>
<td>x-position</td>
</tr>
<tr>
<td>INT</td>
<td>y</td>
<td>y-position</td>
</tr>
<tr>
<td>DOUBLE[]</td>
<td>rgb</td>
<td>RGB triple</td>
</tr>
</tbody>
</table>

5.2.10 PUBLIC setPixelYCC

Description
sets YCbCr triple on a position in a given BUFFEREDIMAGE

Settings
none

Parameters
<table>
<thead>
<tr>
<th>BUFFEREDIMAGE</th>
<th>bim</th>
<th>image</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>x</td>
<td>x-position</td>
</tr>
<tr>
<td>INT</td>
<td>y</td>
<td>y-position</td>
</tr>
<tr>
<td>INT[]</td>
<td>ycc</td>
<td>YCbCr triple</td>
</tr>
</tbody>
</table>
5.2.11 **PUBLIC** setPixelGray

**Description**
sets gray value (integer) on a position in a given BUFFEREDIMAGE

**Settings**
none

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFFEREDIMAGE bim</td>
</tr>
<tr>
<td>INT x</td>
</tr>
<tr>
<td>INT y</td>
</tr>
<tr>
<td>INT gray</td>
</tr>
</tbody>
</table>

5.2.12 **PUBLIC** setPixelGray

**Description**
sets gray value (floating point) on a position in a given BUFFEREDIMAGE

**Settings**
none

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFFEREDIMAGE bim</td>
</tr>
<tr>
<td>INT x</td>
</tr>
<tr>
<td>INT y</td>
</tr>
<tr>
<td>DOUBLE gray</td>
</tr>
</tbody>
</table>

5.2.13 **PUBLIC** cloneImage

**Description**
clones a given BUFFEREDIMAGE

**Settings**
none

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFFEREDIMAGE bim</td>
</tr>
</tbody>
</table>

**Output**
BUFFEREDIMAGE cloned image
5.2.14 **PUBLIC** readImage

**Description**
reads a given file into a **BUFFERedImage**

**Settings**
none

**Parameters**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRING</td>
<td>fileName</td>
<td>filename to read in</td>
</tr>
</tbody>
</table>

**Output**
**BUFFERedImage** resulting image

5.2.15 **PUBLIC** saveImage

**Description**
saves a given **BUFFERedImage** to a given file using a given format

**Settings**
none

**Parameters**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BUFFERedImage</strong></td>
<td>bim</td>
<td>image to save</td>
</tr>
<tr>
<td>STRING</td>
<td>fileName</td>
<td>filename to be used</td>
</tr>
<tr>
<td>STRING</td>
<td>formatName</td>
<td>file-format to be used (jpg, gif, etc.)</td>
</tr>
</tbody>
</table>

**Output**
**BUFFERedImage** resulting image
5.3 PRNG

5.3.1 Description

PRNG provides methods for generating pseudo-random numbers, bytes and booleans. These pseudo-random values are based on an initial-seed and can be re-created by using the same seed again.

5.3.2 Constructor PRNG

Description
constructs a PRNG object with the given initial-seed

Settings
none

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYTE[] seed</td>
<td>initial-seed</td>
</tr>
</tbody>
</table>

5.3.3 Public reset

Description
resets the PRNG to its initial-state

Settings
none

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td></td>
</tr>
</tbody>
</table>

5.3.4 Public nextInt

Description
creates the next pseudo-random number between 0 (inclusive) and a given 'n' (exclusive)

Settings
none

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT n</td>
<td>upper bound for resulting number</td>
</tr>
</tbody>
</table>

Output

INT pseudo-random number between 0 and given 'n'
5.3.5 **PUBLIC nextInt**

**Description**
creates the next pseudo-random number

**Settings**
none

**Parameters**
none

**Output**
INT pseudo-random number

5.3.6 **PUBLIC nextBoolean**

**Description**
creates the next pseudo-random boolean value

**Settings**
none

**Parameters**
none

**Output**
BOOLEAN pseudo-random boolean value

5.3.7 **PUBLIC nextBytes**

**Description**
creates the next pseudo-random byte-array

**Settings**
none

**Parameters**

| INT size | size of byte-array to be filled with pseudo-random values |

**Output**
BYTE[] pseudo-random byte-array
6 Settings

Global settings, parameters and constants. These variables may be used to customize the program's behaviour to individual needs.

Note: some variables have inner dependencies.

- **aesKeySize**: key-size used for AES-encrypted chunks
- **chunkNamePrefix**: filename prefix to be used for encrypted chunks
- **CKP.hd**: default certified key password
- **CKtag**: default label to store certified key under
- **dctCoefficients**: DCT-Coefficients for DCT/IDCT calculations (pre-calculated for dim=8)
- **dim**: dimension of DCT-Blocks
- **frameNamePrefix**: filename prefix to be used for extracted JPEG-frames
- **framesPerChunk**: number of frames to be packed into a single chunk
- **Import.defaultFramesPerSec**: default FPS of converted MJPEG-videos
- **Import.defaultLeadingZeroes**: number of zeroes to pad filenames with
- **Import.defaultOutputDir**: default output directory to store imported videos in (used in fixed-mode of ImportVideo)
- **Import.defaultVideoFile**: default video file to import (used in fixed-mode of ImportVideo)
- **Import.defaultVideoName**: default name to store video under (used in fixed-mode of ImportVideo)
- **payloadBitsPerFrame**: number of payload bits to be embedded per frame
- **SED.edgeScale**: weighting-parameter for detected edge-value (SmoothEdgeDetection)
- **SED.edginess**: threshold for blocks that can be marked without affecting perceptibility (SmoothEdgeDetection)
- **SED.offset**: parameter added to Smooth/Edge-Level (SmoothEdgeDetection)
- **SED.showResult**: gray value to border found blocks with; if set to -1 blocks are NOT bordered (SmoothEdgeDetection)
- **SED.smoothScale**: weighting-parameter for detected smooth-value (SmoothEdgeDetection)
- **sequenceBitsPerFrame**: number of sequence bits to be embedded per frame
**Settings**

- **startCoefficient** ................. coefficient to start marking from (in ZigZag-Order)
- **VCT_Port** ......................... Port used for VideoChunkTransmission
- **Vserver_defaultMessage** ........... default message to be embedded (demonstration purposes)
- **Vserver_defaultPeer** ............... default peer to fill PeerList with (demonstration purposes)
- **Vserver_defaultWMseed** ............ default seed to be used for watermarking (demonstration purposes)
- **Vserver_IP** ........................ Vserver's IP-Address (used in fixed-mode of Vclient)
- **Vserver_Port** ....................... Vserver's Port
- **watermarkKeySize** ................. key-size used for watermarking
- **watermarkStrength** ............... multiplier for applied FridrichPattern (affects detectability of watermark)
- **zigZagOrder** ........................ coordinates of ZigZag-Order (pre-calculated for dim=8)
This project uses **ethemba** as underlying TPM framework for Java.

ethemba allows for convenient high-level access to TPM functionalities. It is based on **jTSS**, a Trusted Software Stack implementing all the TSS layers directly in Java. It is developed and maintained at the Institute for Applied Information Processing and Communication (Institut für angewandte Informationsverarbeitung und Kommunikation, IAIK) at Graz University of Technology (TU Graz).

For further information on ethemba, jTSS and the OpenTC Project visit the following websites:

**ethemba - Ethemba Trusted Host Environment Mainly Based on Attestation**
http://www.ethemba.info

**Trusted Computing for the Java Platform**
http://trustedjava.sourceforge.net

**OpenTC Project**
http://www.opentc.net

**Institute for Applied Information Processing and Communications**
http://www.iaik.tugraz.at