Copyright (C) 2021, Axis Communications AB, Lund, Sweden. All Rights Reserved.

Signed Video feature description

This is a high level description of the *Signed Video* feature. For a correct, detailed and complete description see the source code. A high level description will help understand what the source code aims to accomplish. Therefore, even though this description may have minor flaws, it is good to read before digging into the source code.

What is it?

The feature *Signed Video* adds cryptographic signatures to a captured video as part of the video codec format. The video is, after adding these signatures, protected against manipulations, that is, the authenticity of the video can be validated if needed.

Full tampering coverage is not guaranteed. There are still various tampering scenarios that are not handled and also, authenticity is a matter of trust. For example, a complete authenticity validation of a video is only possible if the user trusts the Public key used for verifying the signature. If the Public key cannot be trusted, the framework will still signal the video as authentic, since validating the Public key is out of the scope.

Summary

A video consists of picture frames displayed at a certain frame rate. If these frames are transmitted or stored for later use and displayed by a third party one would like to be able to validate that they have not been manipulated since the time of signing.

In brief, the principle of signing documents is used, that is, collect information and sign the information using a Private encryption key. Then, packetize the produced signature together with some additional information. For validation, the user can then verify the information by using the signature and the corresponding Public key.

On a high level, *Signed Video* hashes encoded video frames and on a regular basis creates a document representing these hashes and signs that document. This signature, together with the document, is added to the video using Supplementary Enhancement Information (SEI) frames.

Limitations and properties

Signed Video is currently only available for the video codec formats H264 and H265. Therefore, most of the description uses the Network Abstraction Layer (NAL) concept. Note that raw videos are not supported.

Signing frequency. Signing is done upon transition between two Group of Pictures (GOP). For short GOP lengths the time between two GOP transitions may be shorter than the time it takes to perform the signing. Hence, there is a limit on how short GOPs the framework can allow for to be able to sign a video in real-time.

Authenticity level. *Signed Video* supports two levels of authenticity; GOP level and NALU level. For GOP level, all frames between two signatures are treated in one single chunk and if the validation fails, all these

frames are marked as not authentic even if it is due to a lost frame. For NALU level the framework can identify which frames are authentic or not, or even lost. The cost for validating the authenticity of each individual NAL is an increase in bitrate.

Detailed description

As mentioned, the framework currently only supports H264 and H265. These codec formats allow the user to add arbitrary data to a stream through SEI frames of type *user data unregistered*. *Signed Video* puts the produced signatures and additional metadata in such frames. These SEI frames are ignored by the decoder and will therefore not affect the video rendering. One obvious drawback is that it is easy to destroy the signed video and make it unsigned by simply dropping those SEI frames. In some cases this can also be benefitial if, e.g., the user is no longer interested in its authenticity. It is out of scope to protect against lost SEI frames.

All operations are done on the encoded video stream. Each picture frame is split into NALUs Units (NALU) and *Signed Video* operates on these NALUs. NALUs that are not part of a picture frame are ignored. These NALUs are

- SPS/PPS/VPS
- AUD
- SEIs other than Signed Video specific

Note that these can still affect the visual aspect of a video.

Signing a GOP

Without loss of generality, consider three consecutive GOPs each starting with an IDR (I-frame) followed by 4 non-IDRs (P-frames). In text format it would look like `IPPPPIPPPPIPPPP`. The signing information is collected in a SEI frame (`S`) and put just before the picture frame to follow the Access Unit AU format. Each I-frame will trigger a signing procedure and ideally the SEI is generated and available instantaneously and can be attached to the stream as `SIPPPPSIPPPPSIPPPP`.

Each NALU is hashed using SHA-256, but not in a straightforward manner. Since every P-frame directly or indirectly refers to the I-frame starting the GOP they are linked together. Let `h(F)` denote the hash of a frame `F`, and `href = h(I)` is the hash of the first I-frame in a GOP and used as reference. Then each frame in a GOP is hashed according to `hash(F) = h(href, h(F))` where `href` and `h(F)` have been aligned in memory. All hashes are collected in a list and together with some metadata form a `document`, which later will be signed.

To preserve the order of GOPs the I-frame of the next GOP is also include in the list of hashes, that is, `hash(Inext) = h(href, h(Inext))` is added as well.

This `document` is then hashed and signed to produce a signature as

```
signature = sign(h(document))
```

and together with the `document` itself is added to the stream in a SEI, that is, SEI = `document + signature`. After signing, the next GOP is then initiated with a new `href` using the very same I-frame that closed the previous GOP. For the end user to validate the authenticity of a signed video the public key, associated with the private key used when signing, is needed. The *Signed Video Framework* supports

including the public key as part of the metadata. This simplifies validating the authenticity of the video, but requires a separate logic to verify its origin.

GOP level signing

Transmitting the list of hashes can be too expensive in terms of an increased bitrate. The *Signed Video Framework* therefore offer a light version in GOP level as authenticitiy level. Instead of the hash list one single hash to represent the entire document is computed. This single hash represents both the metadata and all the frame hashes, and is implemented recursively. The recursive operation is initialized with a hashed salt `hash(0) = h(salt)`. The next step is to add `href` as `hash(1) = h(hash(0), href)` and the n'th hash becomes `hash(n) = h(hash(n-1), hash(F_n))`, where `F_n` is the frame corresponding to the n'th hash. The recursive hash is finalized with the document hash itself, now without the list of hashes. Hence, it includes the metadata only. This final hash, denoted as a gop hash, is

```
hash(gop) = h(hash(N), hash(document))
```

This hash is then signed and the generated signature. Together with the `document` the signature then forms the SEI = `document + signature` where `signature = sign(hash(gop))`

For NALU level and long GOP lengths, *Signed Video* automatically falls back to GOP level to avoid very large SEI frames.

Metadata

Part from the public key it is possible to add some signer specific information. That information is today locked to the fields

- Hardware ID
- Firmware version (can be used if the *Signed Video Framework* is integrated in another code base)
- Serial No
- Manufacturer (Who is the signer, for example Axis Communication AB)
- Address (Contact information of signer)

SEI format

The framework uses the *user data unregistered* type of SEIs. These are organized as

```
| NALU header | payload size | UUID | payload | stop bit |
```

The UUID is used to put a *Signed Video* identity to the SEI. The payload includes the metadata and the signature, and is serialized in a TLV structure. The payload TLVs are further organized as

```
| NALU header | payload size | UUID | metadata | list of hashes | signature |
stop bit |

| ------------------------ document ------------------------- | signature |
stop bit |
```

By definion the `document` includes everything from the NALU header to the signature tag, hence the entire frame is secured.

Signing in a secure hardware

When signing in hardware the signing itself may take some time and to avoid piling up frames *Signed Video Framework* supports the SEI frames being added at a later stage, but no later than at the next signing request.