ONVIF™
Media Signing Specification

Version 25.12

December, 2025

Copyright © 2022-2025 ONVIF™ All rights reserved.

Recipients of this document may copy, distribute, publish, or display this document so long as this copyright notice, license and disclaimer are retained with all copies of the document. No license is granted to modify this document.

THIS DOCUMENT IS PROVIDED "AS IS," AND THE CORPORATION AND ITS MEMBERS AND THEIR AFFILIATES, MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; OR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

IN NO EVENT WILL THE CORPORATION OR ITS MEMBERS OR THEIR AFFILIATES BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL DAMAGES, ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THIS DOCUMENT, WHETHER OR NOT (1) THE CORPORATION, MEMBERS OR THEIR AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR (2) SUCH DAMAGES WERE REASONABLY FORESEEABLE, AND ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THIS DOCUMENT. THE FOREGOING DISCLAIMER AND LIMITATION ON LIABILITY DO NOT APPLY TO, INVALIDATE, OR LIMIT REPRESENTATIONS AND WARRANTIES MADE BY THE MEMBERS AND THEIR RESPECTIVE AFFILIATES TO THE CORPORATION AND OTHER MEMBERS IN CERTAIN WRITTEN POLICIES OF THE CORPORATION.

CONTENTS

1 Scope

This specification defines how media from ONVIF devices is signed to give a guarantee that the media has not been manipulated since it was transmitted from the device. This specification further defines how an ONVIF client should act at file export to preserve the signing feature, and also how an ONVIF client should validate the authenticity of a signed stream. Audio is outside of scope.

2 Normative references

ITU-T Recommendation, H.264: Advanced video coding for generic audiovisual services
<https://www.itu.int/rec/T-REC-H.264>

ITU-T Recommendation, H.265: High effiency video coding
<https://www.itu.int/rec/T-REC-H.265>

ITU-T Recommendation X.660 (2011) | ISO/IEC 9834-1:2012, Information technology — Procedures for the operation of object identifier registration authorities — Part 1: General procedures and top arcs of the international object identifier tree
<https://www.itu.int/rec/T-REC-X.660-201107-I/en>

Global OID reference database
<http://oid-info.com/>

IETF RFC 6234 - US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)
<http://tools.ietf.org/html/rfc6234>

Reference implementation of this specification
<https://github.com/onvif/signed-media-framework>

3 Terms and Definitions

3.1 Definitions

Document	A document in this specification refers to a codec specific document that includes hashes of all media frames since previous signing occasion together with other metadata. For H.264 and H.265 a document is encoded into a SEI frame.
Certificate	A certificate as used in this specification binds a public key to a subject entity. The certificate is digitally signed by the certificate issuer to allow for verifying its authenticity. In order to verify a certificate, one might need to include one or more intermediate certificates.
Signature	A digital signature or digital signature scheme is a mathematical scheme for demonstrating the authenticity of a digital message or document.

3.2 Abbreviations

AU	Access Unit
GOP	Group Of Pictures
IDR	Instantaneous Decoding Refresh
NAL	Network Abstraction Layer
SEI	Supplementary Enhancement Information
TLV	Type-Length-Value

4 Overview

Media Signing is the process of adding cryptographic signatures to captured media as part of the codec format. By adding these signatures, the media can be protected against manipulation (tampering) and the authenticity can be validated.

The public key, that is used to verify the signatures, is also included in the media, as well as certificates to verify the public key. The CA certificate necessary to verify the certificates is not included and has to be acquired from an external trusted source.

5 Signed Video

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 it is valuable if it is possible to validate that the video has 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 signing key. Then, packetize the produced signature together with the collected information. For complete validation, the user first verifies the certificates with a corresponding CA certificate, followed by Public key verification with the certificate, and finally verifies the signature using the Public key. If successful so far, the transmitted information can then be used to validate the video segment.

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

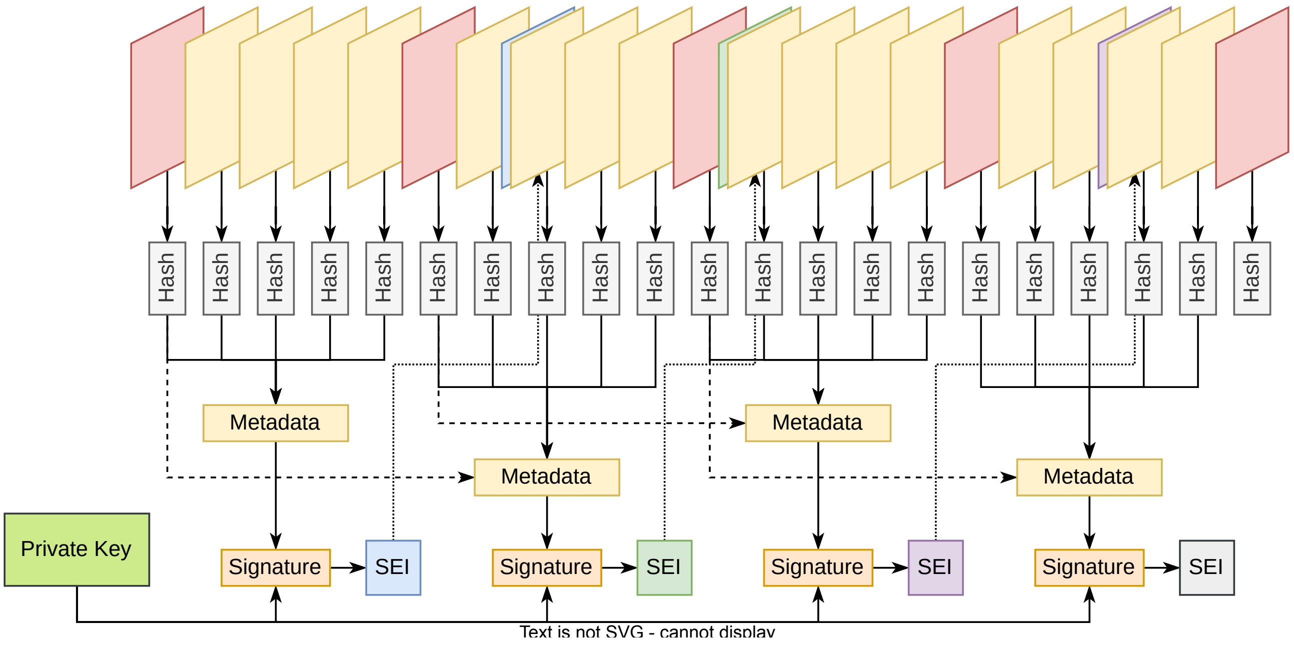

Figure 1: Signed video overview

5.1 Limitations

Codecs. *Signed Video* is defined for the H.264 and H.265 video codec formats, for full details see references. Therefore, most of the description uses the Network Abstraction Layer (NAL) concept. For simplicity, hashing is done on NAL Units instead of full frame data. One frame may be split into multiple NAL Units and therefore generating multiple hashes per frame. Note that raw video is not covered, since there are no means to add signatures within the video format. There exists other codec standards where the same technique can be applied though, but the specification considers H.264 and H.265.

Signing frequency. Signing is done upon transition between two Group of Pictures (GOPs), that is, triggered by an IDR frame. For short GOP lengths, the time between two GOP transitions may be shorter than the time it takes to perform the signing. The standard allows the device to sign multiple GOPs if necessary. This is described in Section 5.5. For very long, or infinite, GOPs the duration between signatures can be too long for practical usage. Therefore, the device is recommended to sign partial GOPs; See Section 5.6.

Frame drops and packet losses. Adding a list of hashes of each NAL Unit combats the problem of lost frames, but cannot solve packet losses unless the affected NAL Unit is completely removed from the stream. To lower the risk of losing packets within a SEI frame, the specification allows the device to add redundancy by transmitting the same SEI multiple times; See Section 5.11 below.

Hashing algorithms. Hashing algorithms, for example those specifed in RFC 6234, are generally used to compute a fixed length hash value from any long input string. Hashing is used in two contexts; 1) to get a short representation of each NAL Unit to put in a list, and 2) as message digest when generating signatures. Note that these two can be different. The same hashing algorithms have to be used in both the device and the client. The device specifies which hashing algorithms were used, which for example limits the ability for the client to put extra security on some videos.

Signing algorithm. The standard specifies the signing algorithm through its OID as definied by the relevant authority or searchable in the global OID reference database. Signing should be done in a trusted environment and there are usually limitations to which signing algorithms are supported. This limits the scope, and the device decides which to use.

Bitrate increase. Adding SEI frames to the encoded video increases the bitrate. The list of NAL Unit hashes is the main contribution. Therefore, it is recommended to consider a hashing algorithm that produces reasonably large hashes, e.g., SHA256. Note that the main security lies in the hash used to generate the signature and not the hash used to represent the NAL Unit. The device has an option to select a low bitrate mode; See Section 5.8.

5.2 Basic rules for the device when signing a video

The procedure of signing a video stream and producing SEI frames, can be described by a set of rules.

- The NAL Unit types below must be hashed

  Picture NAL Units, that is, slices of IDR-, P- and B-frames.

  All SEI frames generated according to this specification, if not including a signature.

- Every GOP must be associated with at least one SEI frame

  Long GOPs should be split into partial GOPs.

- Every partial GOP must be signed

- Every SEI must be linked with the first picture NAL Unit of the previous partial or complete GOP.

  This is necessary to secure the order of frames from tampering.

- The generated SEIs shall be added to the stream in sequence order.

5.3 Detailed description on how the device generates SEI frames including signatures

The H.264 and H.265 codec formats, as specified in referenced documents, allows 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 change the video from signed to unsigned by dropping these SEI frames. In some cases this can also be beneficial. For example, if the user is no longer interested in its authenticity, the SEI frames can be removed and thereby reducing the bitrate. It is out of scope to protect against lost SEI frames. Validating a video where SEIs have been lost or removed will anyhow be marked as not authentic, or unsigned (if no SEIs are present).

All operations are done on the encoded video stream. Each picture frame is split into NAL Units and *Signed Video* operates on these NAL Units. Note that it is not necessarily a one-to-one correspondance between frames and NAL Units. A frame split into slices (multiple NAL Units) will generate multiple hashes. In this specification, unless explicitly stated, hashing a frame means hashing a NAL Unit.

All picture NAL Units shall be hashed. In addition, all SEI frames, generated by ONVIF Media Signing shall also be hashed accordingly, unless the SEI includes a signature.

Some NAL Units have no or limited impact on the visual video and are ignored:

- SPS/PPS/VPS

  The presence of these NAL Units are not consistant. They can for example be sent once in the beginning of a stream and stored at client side to add back when exporting to file. This is a correct process, but affects the authenticity since Signed Video requires the order to be preserved, unless handled separately. There are parts in these NAL Units that can change the video behavior, but not to question its authenticity.

- AUD

  The Access Unit Delimiter (AUD) is simply a NAL Unit header used in Access Unit (AU) video streams, c.f., bytestream. This NAL Unit has no impact on the video and can safely be ignored. By ignoring them, the authenticity is preserved when converting between AU streams and bytestreams.

- SEIs other than Signed Video specific

  There are various types of SEI frames; See the H.264 and H.265 standards. In general, these SEI frames include information that has no visible impact on the video. Further, some of them can correctly be added to, or removed from, the stream at any point in time. This makes it difficult to handle from an authenticity point of view since the presence and order is important. Therefore, all other SEI frames than those described in this specification are ignored.

Note that these frame types can still affect the visual aspect of a video, but are excluded for simplicity. The authenticity of the video is still preserved.

5.4 Signing a GOP

A GOP is defined as all frames between two IDR frames, including the first IDR. Frames between these IDRs are prediction frames and can be either P- or B-frames. Without loss of generality, both of these are throughout the text denoted P-frames. Further, for simplicity IDRs are referred to as I-frames in this context.

Each NAL Unit, except SEI-frames (see below) and those ignored (see above), is hashed in a linked manner. Since every P-frame directly or indirectly refers to the I-frame starting the GOP they are linked together. If the I-frame is modified it indirectly modifies the P-frames. Therefore, each P-frame is hashed together with the dependent I-frame.

SEI-frames are hashed individually. The benefit is that the hash becomes independent of any I-frame. The SEI-frame content may then be verified and trusted even if an I-frame suffer from, for example, packet loss.

Figure 2 illustrates the linking procedure where a frame is encaptured in a single NAL Unit. The hash of the IDR-frame is stored and used to generate a second hash together with the hash of each P-frame in the GOP.

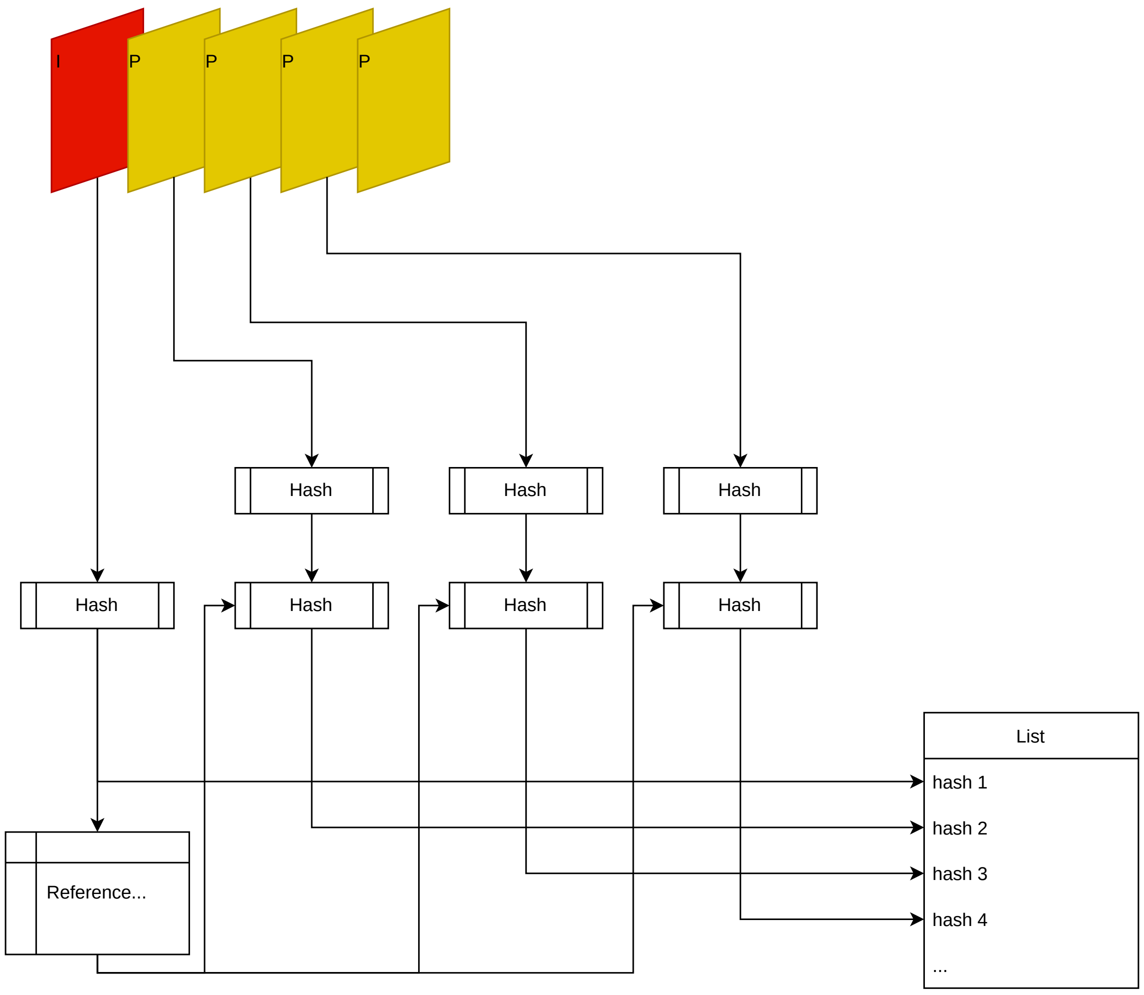

Figure 2: Linked hashes

Figure 3 illustrates the linking procedure where each frame is encaptured in multiple NAL Units. The hash of the primary slice (dark red) of the I-frame is stored as an anchor hash and used to generate a second hash together with any other hash in the GOP. Note that the other NAL Units (non-primary slices) of the I-frame are hashed in this linked manner. In the example in Figure 3, there is one primary slice (dark red) and one non-primary slice NAL Unit (light red) illustrating the linking procedure where a frame is encaptured in multiple NAL Units. The hash of the primary slice of the I-frame is stored and used to generate a second hash together with any other hash in the GOP. Note that the other NAL Units of the I-frame are also hashed in this linked manner.

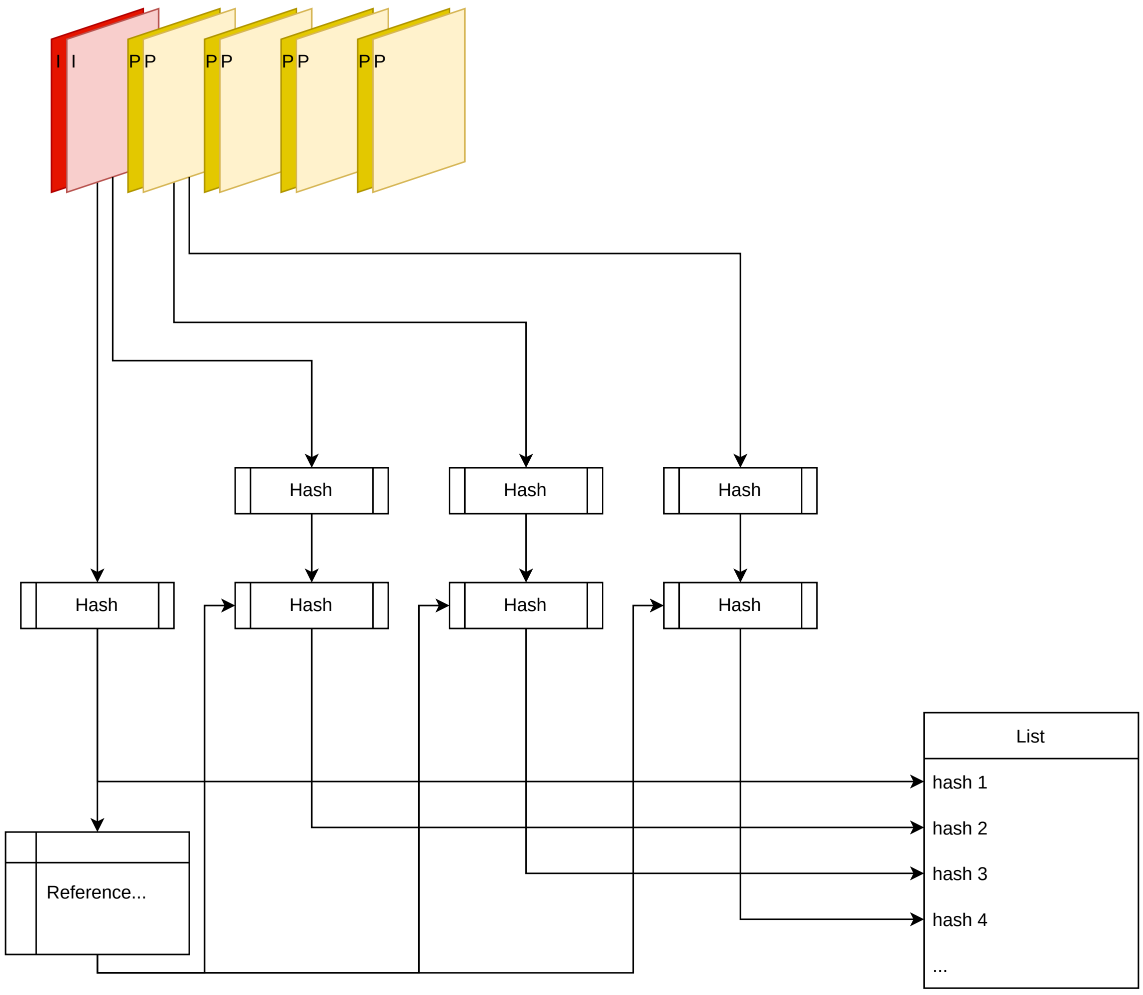

Figure 3: Linked hashes, multiple NAL Units per frame

Let `H(F)` denote the hash of a NAL Unit `F`, and `h_anch = H(I)` is the hash of the leading I-frame in a GOP and used as anchor (linked hash). Then each frame in a GOP is hashed according to `hash(F) = H(h_anch, H(F))`, where `h_anch` and `H(F)` shall be hashed as if they were aligned in memory. All hashes are collected in a list and together with some metadata forms a `document`, which later will be signed.

All NAL Unit data, from the NAL Unit header to the last byte, including the stop bit, shall be hashed. The start code is excluded when hashing.

If the I-frame is sliced into multiple NAL Units, the hash of the primary slice shall be used as `h_anch = H(Iprimaryslice)`. The other NAL Units of the I-frame are hashed in the linked manner described above.

The NAL Units shall be hashed in the same order as the encoder emits them, that is, in decoding order. For example, consider 4 consecutive images with data D1, D2, D3, D4. Let the encoder use a GOP structure with two B-frames, which in presentation order would be I-B-B-P. The order out from the encoder becomes I1-P4-B2-B3, which also is the order of which they shall be hashed and put in the list of hashes [h_ach, hash(P4), hash(B2), hash(B3)].

A hash (`gophash`) of the list of hashes is added to metadata, `gophash = H(h_anch, hash(NALUnit1), hash(NALUnit2), ..., hash(NALUnitN))`. The `gophash` simplifies validation, since verifying each hash in the list of hashes is only necessary when verification of the `gophash` fails. Another benefit is that the bytestream is prepared for the future, where a low-bitrate alternative can be added by omitting the list of hashes.

To preserve the order of GOPs the hash of the I-frame of the previous GOP is also added to metadata, that is, `hash(Iprevious) = h_anch_previous`. For the very first GOP a static and pre-defined hash shall be used; See the TLV section below.

This `document` is then signed to produce a signature as `signature = sign(H(document))` and together with the `document` itself added to the stream in a SEI, that is, SEI = `document + signature`. The next GOP is initiated with a new `h_anch` using the very same I-frame that triggered the end of current GOP.

Figure 4 illustrates the overall signing procedure.

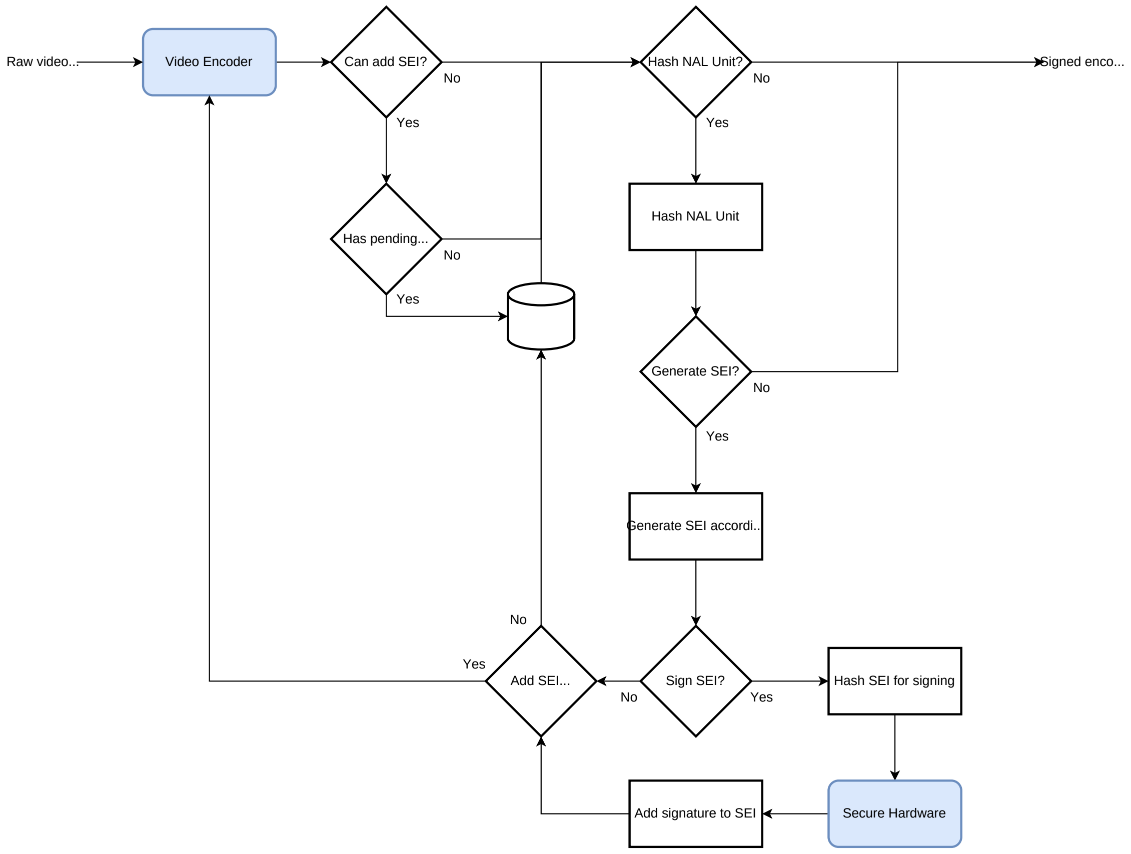

Figure 4: Signing Flow Chart

5.4.1 Example

The following figures show a typical example where each GOP is signed. In this example the GOP is 8 frames long where each I-frame (grey box with letter 'I') marks the start of a GOP, followed by 7 P-frames (white box with letter 'P'). Each I-frame triggers a signing procedure by generating a SEI (Supplemant Enhancement Information) frame of type "user data unregistered". Information, such as hashes from the frames, of the previous GOP is put in this SEI. The SEIs are colored yellow before being signed and green afterwards. SEIs (yellow) are hashed and sent to the signing hardware and added to the stream after signing (now green). The solid lines mark the frames that are signed/secured and the dashed line which frame the GOP is linked with to secure the order of GOPs.

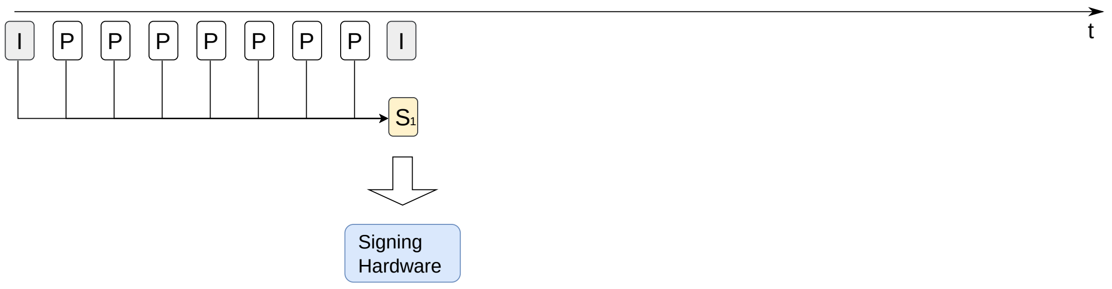

Figure 5: A generated SEI is hashed and sent to the signing hardware.

Signing generally takes time. When ready, the signature is added to the SEI and injected attached to a frame into the stream.

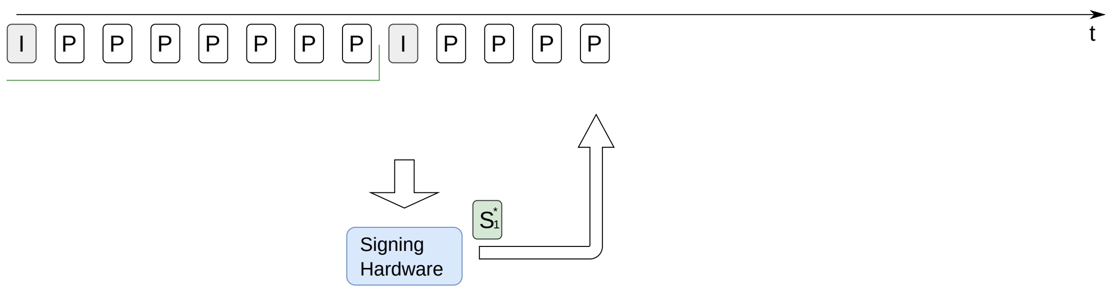

Figure 6: Signature is injected into the stream.

A new I-frame triggers the next signing procedure.

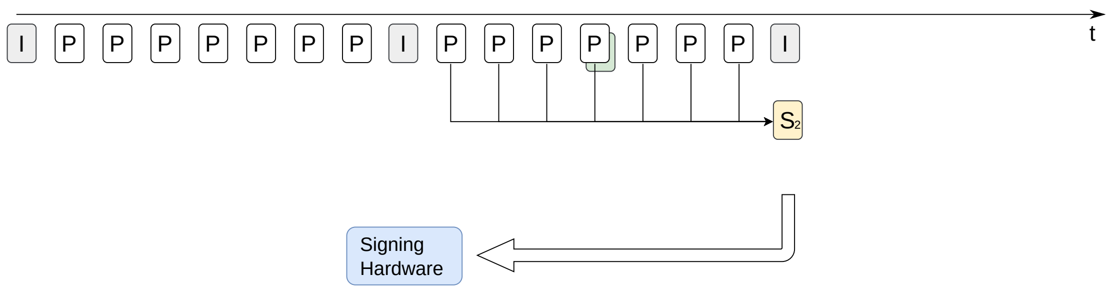

Figure 7: Next I-frame triggers next signing.

When second GOP is signed, the signature is added to the SEI and injected attached to a frame into the stream.

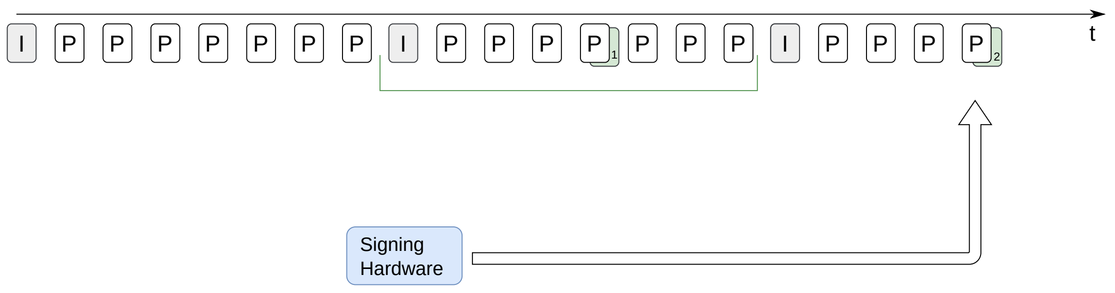

Figure 8: Injecting second signed SEI.

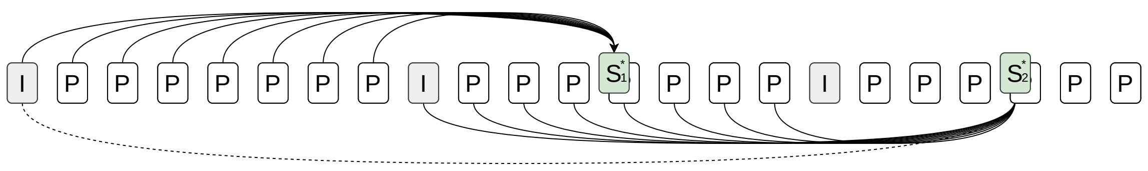

Figure 9: An overview of which frames are secured by which SEI.

5.5 Signing multiple GOPs

As mentioned in the Limitation Section above, there are situations when signing each GOP is not feasible in real-time. The device has an option to decide to sign every X:th GOP to keep the stream signed.

Generate a SEI for every GOP according to the Section above, but sign only every X:th GOP. That is, every X:th SEI includes a Signature TLV.

All, by this specification, generated SEI frames without a signature must be hashed and added to the list of NAL Unit hashes. Note that they have to be hashed in the same order as they appear in the stream.

The SEI frames shall be added in the exact order they are generated, regardless of their readiness. For example, a SEI frame without a signature that is ready before the previous signed SEI is, cannot be added until the signed SEI is completed.

5.5.1 Example

The following example figures show how multiple short GOPs can be signed. The yellow boxes are used to illustrate unsigned SEIs and green boxes are signed SEIs.

Figure 10 and Figure 11 show how an I-frame starts a new GOP and a SEI representing the previous GOP is generated. These SEIs are not signed and can be injected to the stream instantly.

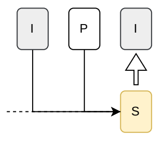

Figure 10: Unsigned SEI of first I-frame.

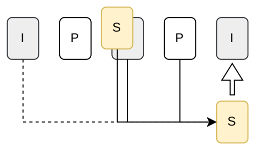

Figure 11: Unsigned SEI of second I-frame.

Figure 10 shows that after generating the SEI representing GOP 3, it is sent to the signing hardware.

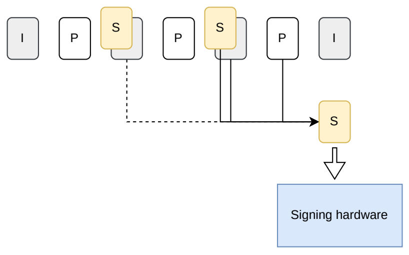

Figure 12: Signing SEI of GOP 3.

While the third SEI is being signed, a SEI representing GOP 4 is generated and waiting to be injected to the stream as shown in Figure 13.

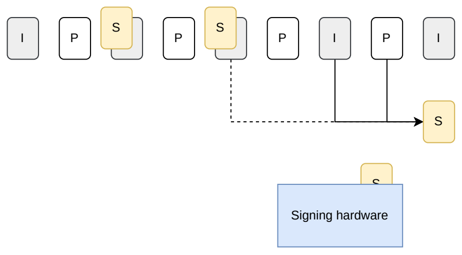

Figure 13: Pending injection of SEI representing GOP 4.

When the signature is ready it is added to the corresponding SEI and injected to the stream as shown in Figure 14.

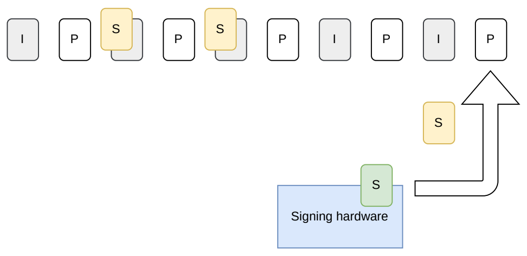

Figure 14: In order injection of signed SEI.

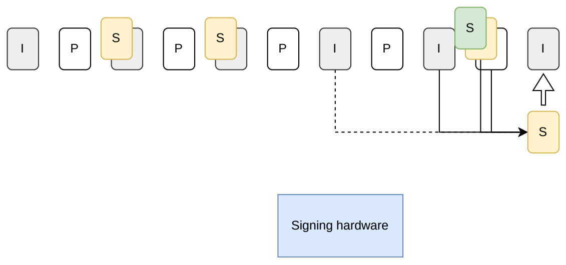

Figure 15: Signing multiple GOPs 6

Finally Figure 16 and Figure 17 provide an overview of which frames are secured by which SEI. In the case of Figure 16 the SEIs have been prepended to the same frame that triggered the SEI while in the case of Figure 17 the SEIs have been added to the stream on the next available instance.

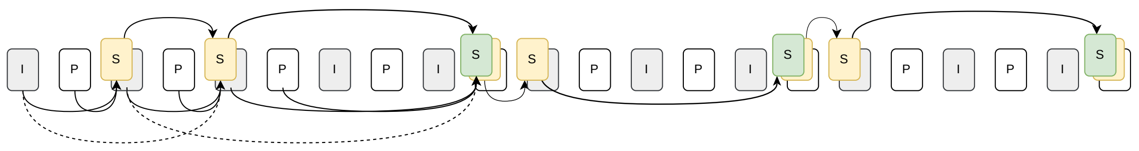

Figure 16: Overview when SEIs are prepended.

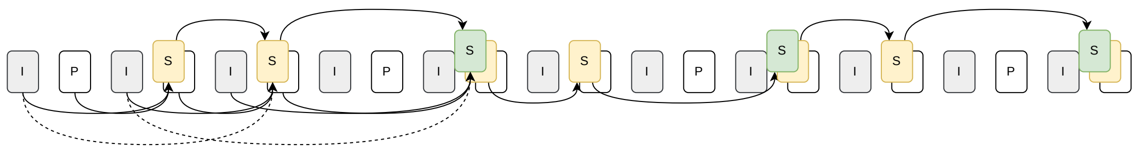

Figure 17: Overview when SEIs are added on the next available instance.

5.6 Signing partial GOPs

As mentioned in the Limitation Section above, very long, or even infinite, GOPs puts the validation side in an unacceptable situation since the duration between signatures is far to long. The device has an option to decide to force signing if the GOP has not ended in a reasonable time. It is recommended to split a GOP into partial GOPs if the GOP duration is longer than 5 seconds.

Generate a SEI for every partial GOP according to the Section above and sign it.

Note that hashing NAL Units use the same anchor hash across partial GOPs. The anchor hash is only reset upon an IDR, that is, when there is a new GOP.

5.6.1 Example

The following example figures shows how long GOPs can be signed by using multiple SEIs.

In Figure 18 the number of NAL Units since the I-frame reached 7 which generates a first SEI that is hashed and sent to the signing hardware. SEI of partial GOP 1

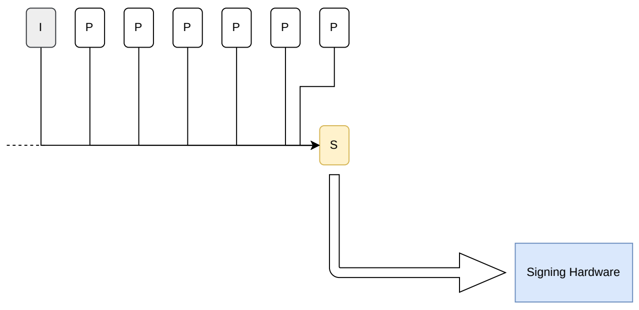

Figure 18: First SEI of partial GOP

In Figure 19 the signature is ready and added to the SEI (green) which is injected to the stream.

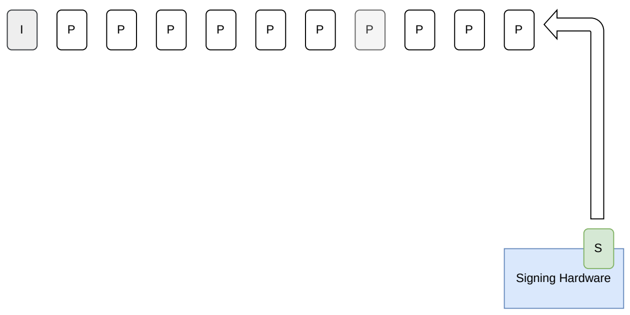

Figure 19: Insert signed partial SEI

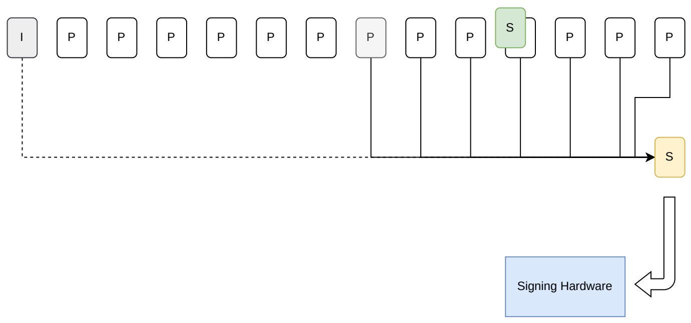

Figure 20: SEI of partial GOP 2

Figure 21 shows that an I-frame triggers signing. A third SEI is generated that is hashed and sent to the signing hardware. SEI of partial GOP 3 put in queue for signing.

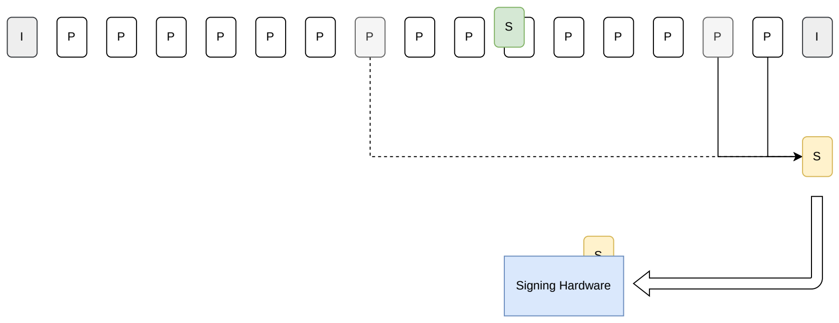

Figure 21: I-Frame triggers SEI signing

When the signature of the second SEI is ready, it is added to the corresponding SEI (green) and injected to the stream as shown in Figure 22.

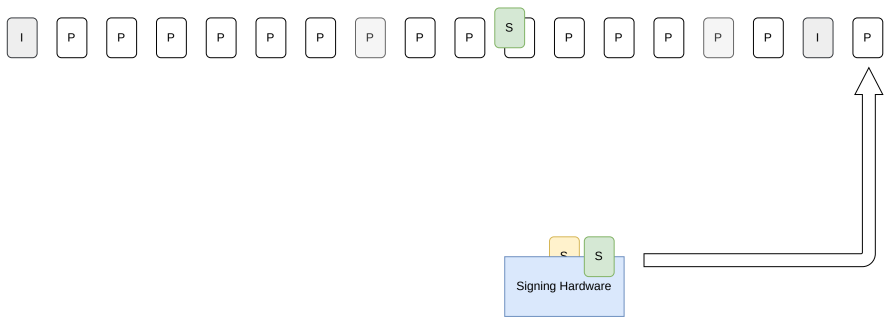

Figure 22: SEI of partial GOP 2 is injected to the stream.

Figure 21 shows that the signature of the third SEI is now ready. It is added to the corresponding SEI (green) and injected to the stream.

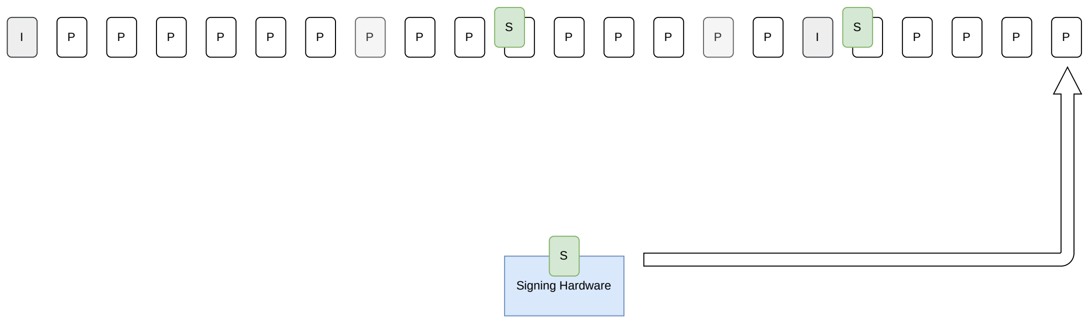

Figure 23: SEI of partial GOP 3 is injected to the stream.

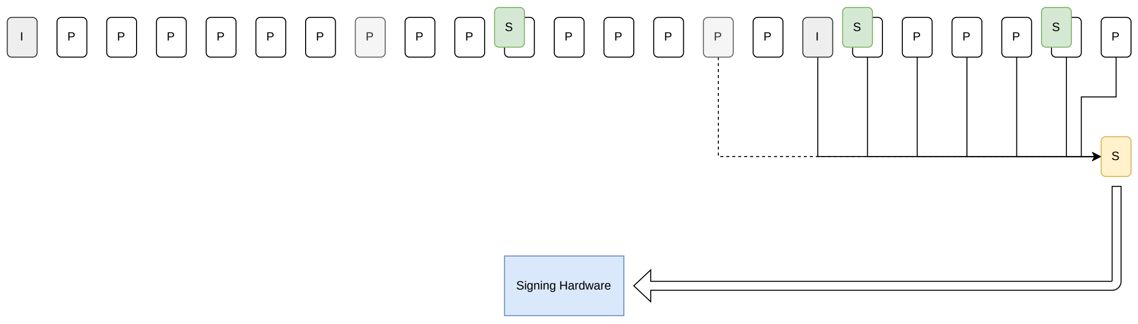

Figure 24: SEI of partial GOP 4.

Finally Figure 25 provides an overview of which frames are secured by which SEI.

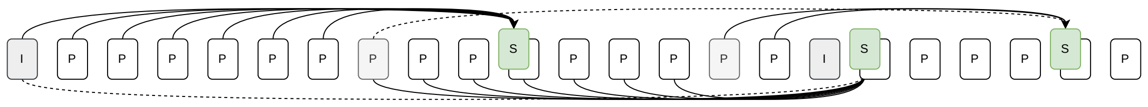

Figure 25: Overview of which frames are secured by which SEI.

5.7 SEI format

SEIs of type *user data unregistered* are used. These are organized as:

Table 1: User data unregistered SEI

NAL Unit header	payload size	UUID	payload	stop bit

The UUID is used to put an *ONVIF Signed Video* identity to the SEI. UUID for Signed Video is `005bc93f-2d71-5e95-ada4-796f90877a6f`. The payload includes the metadata, the list of hashes and the signature, and is serialized in a TLV structure. The order of the added TLV tags is arbitrary except the signature tag which by definition has to come last. By definition the `document` includes everything from the NAL Unit header (inclusive) to the signature tag (exclusive), hence the entire frame is secured.

Table 2: Signed video SEI

document							
NAL Unit header	payload size	UUID	reserved byte	metadata TLVs	list of hashes TLV	signature TLV	stop bit

5.7.1 The reserved byte

Right after the UUID, and before the TLV data, is a reserved byte. This byte represents bit flags of information that cannot be put in the TLV structure.

The first bit shall be set to 1 if this is a certificate SEI.

The second bit shall be set to 1 if emulation prevention was applied **before** this SEI (the document) was hashed for signing.

The last bit shall be set to 0. If, in the future, 7 bits is not enough to signal settings outside the TLV structure, this bit will be used to signal more reserved bytes.

5.7.2 Start of stream

Large data blobs like certificates are preferably sent only once. This "only need once" information can be put in a separate certificate SEI, with the corresponding bit in the reserved byte set to 1, and signed. This self-signed SEI separates the certificate SEI from the video stream and can be copied and added to any exported material.

If the video is signed with a user provisioned signing key, this certificate SEI shall be signed with the manifacturer provisioned signing key. Note that both certificates of Public keys have to be included in the certificate SEI.

5.7.3 TLV tags

Following is a list of possible TLV tags that can be part of a SEI payload. The length field of some of these tags can be represented with only one byte, but for ease of use all tags use 2 bytes for representing the length.

Table 3: TLV tags

Tag	#	Freq	Description
General GOP information	1	Every document	General information like GOP number
Hash list	2	Every document	List of NAL Unit hashes
Signature	3	Every document	The signature of the document
Cryptographic information	4	Once or every document	Cryptographic information such as hashing and signing algorithms
ONVIF vendor information	5	Once or at any arbitrary cadence	Information about the device and vendor

Tag	#	Freq	Description
Certificate chain	6	Once or at any arbitrary cadence	The device video signing key certificate chain. The leaf certificate includes the Public key necessary to verify signatures.
Arbitrary data	7	Optional	Additional data can be put here as a vendor specific blob.

5.7.4 General GOP information

A mandatory field that includes GOP specific information like a GOP counter, a timestamp of the I-frame and the GOP-hash. If this TLV is in a SEI associated with a partial GOP, the timestamp of the first NAL Unit is used.

- Tag version (1 byte)

  Version = 2

- Specification version (3 byte)

  The major and minor bytes shall be set to the specification revision, e.g., [25, 12].

  The patch byte is reserved by the open-source reference implementation code (github.com/onvif/media-signing-framework). Should be set to 0 if not using media-signing-framework.

- Signing triggered by partial-GOP (1 byte)

  This byte is set to 1 if signing was triggered by a partial GOP, that is, before the end of the GOP was reached. All other SEIs have this byte set to 0. For example, if a very long GOP is split into three signings, the first two SEIs will have this byte set to 1 and the last SEI has this byte set to 0.

- The UTC (8 bytes) based time represented by the number of 100-nanosecond intervals since January 1, 1601 of the frame leading the (partial) GOP.

  The start timestamp value is a 64 bit parameter and the 8 bytes shall be written with little endian.

- Only present from tag version 2.

  The UTC (8 bytes) based time represented by the number of 100-nanosecond intervals since January 1, 1601 of the frame leading the upcoming (partial) GOP.

  The end timestamp value is a 64 bit parameter and the 8 bytes shall be written with little endian.

  The duration of the signed (partial) GOP can now be calculated as duration = "end timestamp" - "start timestamp". Note that the end timestamp is exclusive.

- GOP counter (4 bytes)

  A running counter that shall increment by 1 for each GOP. This counter is a helper during validation when detecting lost SEIs and syncing SEIs with the associated GOP.

  The counter value is a 32 bit parameter and the 4 bytes shall be written with little endian.

  Note that the counter is only incremented upon transition to a new GOP and not for each partial-GOP.

- Number of NAL Units represented by partial-GOP hash (2 bytes)

  The GOP-hash added below is by definition a hash of NAL Unit hashes. For simplified validation knowledge of the number of NAL Units used shall be added, represented by an unsigned 16 bit value.

- (Partial) GOP hash (`gophash`) (h bytes)

The hash of the list of NAL Unit hashes. This hash is used to verify the entire GOP in one operation. Upon failure, the list of hashes, if present, is used to detect missing NAL Units. Number of bytes depends on the hash function used.

- Hash of the first hashable NAL Unit in the previous (partial) GOP (h bytes)

  In most cases this is the hash of the linked I-frame, that is, the I-frame leading the previously signed GOP (`h_anch_previous`). Number of bytes depends on the hash function used.

  For the first signed GOP these `h` bytes shall be set to `0x00`.

It is not necessary to specify `h` in this tag. The validation side can compute h since the total tag length is known.
`h = (length - 13) / 2`

5.7.5 Cryptographic information

This mandatory field contains information to perform cryptographic operations such as hashing and signing.

- Tag version (1 byte)

  Version = 1

- The size of the serialized OID of the hash algorithm (1 byte)
- The OID, in serialized format of the hash algorithm used to hash NAL Units (variable byte)
- Signing algorithm defined by its OID and specified here in its serialized form (variable byte)

  Not needed for EC and can therefore be excluded.

- The cryptographic size of RSA (2 bytes)

  Set to zero if RSA is not used

Note that it is not necessary for the device to support all alternatives. The device makes the decision which combination to use.

5.7.6 ONVIF vendor info

Includes information that identifies the product and vendor. The following fields are supported. These fields are only for easy parsing by a player. The true device information is present in the certificate.

- Tag version (1 byte)

  Version = 1

- Size of FW version (1 byte)

  The size of the FW version as a character string, excluding termination character. If no FW version is written, set size to 0.

- Firmware version
- Size of Serial Number (1 byte)

  The size of the Serial Number as a character string, excluding termination character. If no Serial Number is written, set size to 0.

- Serial Number or similar
- Size of Manufacturer (1 byte)

The size of the Manufacturer as a character string, excluding termination character. If no Manufacturer is written, set size to 0.

- Manufacturer (who is the signer)

5.7.7 Hash list

This optional field contains the hash list, for a complete GOP; I- through P- and B-frames.

The presence of this field is controlled by the Low Bitrate Mode; See Section 5.8.

- Tag version (1 byte)

  Version = 1

- All hashes aligned byte by byte (N*H bytes)

5.7.8 Signature

This mandatory field contains the Signature of the document.

- Tag version (1 byte)

  Version = 1

- Actual size of signature (2 bytes)

  This is the written size, since not all signing algorithms generate a fixed size.

- The signature (max signature size bytes)

  A fixed size is reserved to be able to determine the payload size of the SEI. This makes the total size of the TLV tag fixed, hence known in advance. A known size is necessary to write the SEI payload size in advance and then hash the SEI and generate the message digest for signing.

5.7.9 Certificate chain

This field contains the certificate chain necessary to verify the public key.

The leaf certificate includes the Public key, and upon successful verification can safely be extracted to use when verifying the signatures.

Note that the certificate chain must not include the CA certificate.

Note that this tag is mandatory in the certificate SEI and optional thereafter. If sent at stream start, it has to be part of a SEI marked with the certificate SEI flag.

- Tag version (1 byte)

  Version = 1

- User provisioned key (1 byte)

  Set this byte to 0x01 if the certificate chain in this TLV is a user provisioned certificate chain.

- The certificate chain in PEM format (variable bytes)

5.7.10 Arbitrary data

This optional field contains Abritrary data that the device vendor want to be included. There is no guarantee that the data is reasonable or readable by anyone not previously knowledgeable in the area.

The data tag can be added to any SEI described in this specification or to a SEI of its own.

This tag can be useful to add vendor or device specific information and get it signed.

5.8 Low Bitrate Mode

As mentioned in the Limitation Section 5.1 above, the SEI frames can become large due to the hash list. To lower the bitrate overhead, the device has the option to skip the Hash list tag; See Section 5.7.7. Skipping the Hash list tag is referred to as the Low Bitrate Mode.

The Low Bitrate Mode affects the authentication. It is no longer possible to identify individual invalid and/or missing frames, since individual hashes are not transmitted. Instead, validation is based on the (partial) GOP hash present in the general GOP information tag; See Section 5.7.4. For a correctly verified (partial) GOP hash, all NAL Units now have to be present and correct.

5.9 How to generate the first certificate SEI

- Write SEI NAL Unit header

  H.264: `0x06 0x05`

  H.265: `0x4e 0x01 0x05`

- Determine the total payload size, excluding potential emulation prevention bytes. The total payload includes the UUID (16 bytes) and all TLV data. The final stop-bit byte is excluded from the payload size.

- Write the payload size according to the codec standard.

```
while (payload_size >= 0xff) {
    *nal_ptr++ = 0xff;
    payload_size -= 0xff;
}
*nal_ptr++ = payload_size;
```

- Write the UUID

```
0x00 0x5b 0xc9 0x3f 0x2d 0x71 0x5e 0x95 0xad 0xa4 0x79 0x6f 0x90 0x87 0x7a 0x6f
```

- Write the reserved byte with the certificate SEI flag set; `0x80`.

  If emulation prevention will be applied before hashing the SEI for signing, also set the emulation prevention bit; `0xc0`.

- Write all the TLV data except the signature since it by definition cannot be written yet.

  The mandatory tags are the Cryptographic information and the Certificate chain.

  If the video will be signed with a user provisioned signing key write an additional TLV data for the user provisioned certificate chain.

- If the emulation prevention bit is set in the reserved byte, apply emulation prevention on the written data so far.

- Hash the written data from NAL Unit header to the last written TLV data. Denote this "document hash".

- Sign the document hash using the manufacturer provisioned Private signing key, stored in a trusted environment.

- Write the signature data to the Signature tag when signing has been completed.

- Write a stop bit byte; 0x80.

5.10 How to generate a SEI signing a GOP

- Write SEI NAL Unit header

  H.264: `0x06 0x05`

  H.265: `0x4e 0x01 0x05`

- Determine the total payload size, excluding emulation prevention bytes. The total payload includes the UUID (16 bytes) and all TLV data.

- Write the payload size according to the codec standard.

  ```
  while (payload_size >= 0xff) {
      *nal_ptr++ = 0xff;
      payload_size -= 0xff;
  }
  *nal_ptr++ = payload_size;
  ```

- Write the UUID

  ```
  0x00 0x5b 0xc9 0x3f 0x2d 0x71 0x5e 0x95 0xad 0xa4 0x79 0x6f 0x90 0x87 0x7a 0x6f
  ```

- Write the reserved byte without the certificate SEI flag set; `0x00`.

  If emulation prevention will be applied before hashing the SEI for signing, set the emulation prevention bit; `0x40`.

- Write all the TLV data except the signature since it by definition cannot be written yet.

- If the emulation prevention bit is set in the reserved byte, apply emulation preventionon the written data so far.

- Hash the written data from NAL Unit header to the last written TLV data. Denote this "document hash"

- Sign the document hash in a secure environment using the Private signing key.

- Write the signature data to the Signature tag when signing has been completed.

- Write a stop bit byte; 0x80.

5.11 How to add a generated SEI to a video stream in a device

Without loss of generality, consider three consecutive GOPs each starting with an I-frame followed by 4 non-IDRs (P- or B-frames). In text format it would look like `IPPPPIPPPPIPPPP`. The signing information is collected in a SEI frame (`S`). The SEI should be added to the video stream such that it follows the Access Unit (AU) format, i.e., it has to prepend all picture NAL Units in an AU. 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`.

In practice signing in secure hardware takes time and blocking the video stream while waiting adds jitter to the video stream. The device therefore has the option to add it to the video stream later, when signing has been completed. In such a case the text-ified stream can look like `IPSPPPISPPPPIPPSPP`.

If the device decides to add redundancy by transmitting the SEI twice the text-ified stream would look like `IPSSPPPISSPPPPIPPSSPP`.

5.12 Handling SetSynchronizationPoint

When the device recieves a SetSynchronizationPoint command, as specified in the media2 specification, the device shall transmit a new instance of the certificate SEI (if the device apply the principle of certificate SEIs).

Further, the device should start a new SEI signing segment in order for the video to be authenticatable from the synchronization point forward.

Annex A. Revision History

Rev.	Date	Editor	Changes
24.12	Dec-2024	Björn Völcker, Fredrik Svensson, Axel Keskikangas	First release
25.06	Jun-2025	Björn Völcker	Clarify the difference between Firmware and Software version.
25.12	Dec-2025	Björn Völcker	Added a second timestamp to bitstream and removed specification version. Requires SEIs to be added in sequential order.

Annex B. Client considerations (informative)

B.1 Minimum requirements for client handling of a signed video stream

Attached to an AU of a signed video stream may be a certificate SEI including all information that is only sent once; The certificate SEI bit is set. This SEI should be stored for later use. Further, when exporting a video segment to file, this certificate SEI shall be added to the first AU of that recording.

The client shall not remove any SEI-frames from the stream. It is possible to move them to the AU including the I-frame of the following GOP the particular SEI did sign.

B.2 Client side validation

Validation is done (partial) GOP by (partial) GOP, just like the signing side signs (partial) GOP by (partial) GOP. The validation has to be aborted upon any failing step below.

- Get the manufacturer CA certificate from an external source.
- Extract the manufacturer Certificates from the SEI.
- Verify the extracted certificates with the CA certificate.
- Extract the manufacturer Public Key from the SEI.
- Verify the signature of the SEI with the extracted Public Key.
- If a user provisioned signing key was used, get the user CA certificate from an external source.

  Extract the user provisioned Certificate from the SEI.

  Verify the extracted user provisioned certificates with the CA certificate.

  Extract the user provisioned Public Key from the SEI.

  Verify the signature of the SEI with the extracted user provisioned Public Key.

  If the SEI is a certificate SEI, instead verify the signature of the certificate SEI with the extracted manufacturer Public Key.
- Find the SEI associated with the GOP that is to be validated.
- Verify the signature of the SEI, following the description above.
- Hash all NAL Units of the GOP in the same way as is described for the device part above.
- Verify the computed GOP-hash against corresponding GOP-hash in the SEI.

  If successful, all NAL Units are validated as authentic.

  Upon failure, verify each computed hash against corresponding hash in the hash list if present in the SEI.
- The client shall signal AUTHENTIC if all computed hashes are verified successfully against the hashes in the SEI, and the order is preserved and no NAL Units are missing.

  If any missing NAL Unit is detected the client shall signal AUTHENTIC WITH MISSING NAL UNITS.

  If no SEIs are present the client shall signal NOT SIGNED.

If none of the above applies the video is not authentic and the client shall signal NOT AUTHENTIC.

B.3 Certificate validation

In order to validate the certificate, one follows the certificate chain from the leaf certificate to the root certificate according to the x.509 standard. Since Onvif does not provide a trusted certificate store the client implementor and/or the end user must install trusted root CA certificates in the client. Typically one also need the intermediate CA certificate which is also provided by the device. When verifying old video, it may happen that the certificates have expired as compared to the current date. However, the verification concerns the date that the video was created as such the date should be validated against the claimed recording date rather than the current date.

B.4 Exporting a Signed Media

In order to be able to verify an exported segment of Signed Media, the client must include the certificate SEI added in the beginning of the media stream. This can be combined with the Onvif Export File Format in order to provide layered video authentication, the media signing covers authenticity in the media layer where as the Export File Format provides validity on the container level. Even if this Signed Media specification is used, the Onvif Export File Format is still valuable as it provides more information of the provenance of the video as it allows the user to tell the story of the path of the video.

B.5 Dangling End

One special consideration is the possbility of dangling ends, i.e., that the last part of the stream or exported file has no SEI frame associated with it. There can be a number of reasons for this happening such as:

- The stream was ended before the last SEI was signed and transmitted
- The file export did not include the associated SEI
- Other implementation specific issues

When exporting a recording, or trying to verify a dropped videostream, this means that the end of the video may not be covered by the verification information in the current segment.

Annex C. Bibliography

Alliance for Open Media | AV1 Bitstream & Decoding Process Specification
<https://aomedia.org/av1/specification/>

IETF RFC 3447 Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1
<https://tools.ietf.org/rfc/rfc3447.txt>