

# Axis Dome Camera - Video Authentication

Mohammed Rahman

November 26, 2024

## Contents

<b>1</b>	<b>Setup</b>	<b>2</b>
<b>2</b>	<b>Network Storage</b>	<b>2</b>
<b>3</b>	<b>Video Signing</b>	<b>3</b>
<b>4</b>	<b>Framework Dependencies</b>	<b>3</b>
4.1	Meson and ninja . . . . .	3
4.2	OpenSSL . . . . .	3
4.3	libcheck . . . . .	4
4.4	GStreamer . . . . .	4
<b>5</b>	<b>Framework</b>	<b>4</b>
<b>6</b>	<b>Signer</b>	<b>5</b>
<b>7</b>	<b>Validator</b>	<b>5</b>
<b>8</b>	<b>Product Information</b>	<b>6</b>
<b>9</b>	<b>Public Key Validation</b>	<b>8</b>
9.1	Validation and Verification Process . . . . .	8
9.2	Ensuring Secure Public Key Validation . . . . .	9
<b>10</b>	<b>Findings</b>	<b>10</b>
10.1	Validating Non-Tampered Camera Footage . . . . .	10
10.2	Validating Tampered Camera Footage . . . . .	11

# 1 Setup

The setup documented follows the steps listed on the [datasheet](#). A Windows-based system was used to locate the IP address of the Axis Dome camera and setup shared storage, while a Linux-based system was used for the rest of the project.

The camera is currently configured with factory settings, including the default IP address. To locate the IP address, use either AXIS Utility or the Axis Device Manager. In this demonstration, AXIS Utility was used. When AXIS IP Utility is opened and the camera is connected, it should automatically detect the camera. The camera's IP address will be displayed on the left-hand side. Copy this address into a browser to access the camera software.

The details for the account are:

**User:** root

**Password:** Electronic1

To turn on video signing from this window, navigate to **Video** → **Stream** → **General**.

# 2 Network Storage

There are two types of storage available: network storage and SD cards. The former was chosen.

First, a shared folder will need to be created on the Windows machine (this can be done on Linux too), which can be done by following this [guide](#). Then, obtain the IP address of the machine hosting the shared folder. This can be done with:

**Windows:** ipconfig

**Linux:** ip a

Navigate to **System** → **Storage** → **Add Network Storage**. Enter the IP address of the host machine, along with the host's username, password, and the path to the shared folder. Leave all other settings at their default values.

### 3 Video Signing

AXIS cameras can utilise [AXIS Edge Vault](#) to sign videos. Ensure that the cameras in use have up-to-date firmware to enable video signing, as cameras straight from the box may not support this functionality. The process of video signing is outlined on pages 13 to 16 of the [datasheet](#).

## 4 Framework Dependencies

The signed-video-framework requires several dependencies, which are listed below and can be found on their [GitHub](#):

- meson
- ninja
- OpenSSL
- libcheck

To run the two applications, `signer` and `validator`, `GStreamer` will also be needed.

### 4.1 Meson and ninja

To install both `meson` and `ninja`, use the following commands:

```
$ pip install meson
$ pip install ninja
```

### 4.2 OpenSSL

To install `OpenSSL`, use the following commands:

```
$ git clone https://github.com/openssl/openssl
$ cd path/to/cloned/repo
$ ./Configure
$ make
$ make test
```

If the tests do not pass, you may need to run the commands by prepending `sudo`.

### 4.3 libcheck

To install libcheck, run the following command:

```
$ sudo apt-get install check
```

### 4.4 GStreamer

To install GStreamer, run the following command:

```
$ apt-get install libgstreamer1.0-dev  
libgstreamer-plugins-base1.0-dev  
libgstreamer-plugins-bad1.0-dev  
gstreamer1.0-plugins-base  
gstreamer1.0-plugins-good  
gstreamer1.0-plugins-bad  
gstreamer1.0-plugins-ugly  
gstreamer1.0-libav  
gstreamer1.0-tools  
gstreamer1.0-x  
gstreamer1.0-alsa  
gstreamer1.0-gl  
gstreamer1.0-gtk3  
gstreamer1.0-qt5  
gstreamer1.0-pulseaudio
```

## 5 Framework

Both the [signed-video-framework](#) and [signed-video-framework-examples](#) repositories were cloned under the same parent directory. All subsequent commands in the framework, as well as those for the `validator` and `signer` sections, were run in the parent directory.

The framework supports both H.264 and H.265 encoding. Depending on the encoding type, the appropriate format should be used in the signer and validator commands.

To compile the framework to the run the two apps, use the following commands:

```
$ meson --prefix $PWD/local_install signed-video-framework
```

```
build
$ ninja -C build
$ meson install -C build
$ ninja -C build test
```

The unit tests should all pass if done correctly.

## 6 Signer

The commands listed below used to compile the validator can also be found on the [AXIS Signer GitHub](#).

If the camera does not automatically perform the signing, the signer step can be done manually. To do this, you must first compile the signer by following the commands provided below:

```
$ export GST_PLUGIN_PATH=$PWD/my_installs
$ meson --prefix $PWD/my_installs -Dsigner=true signed-
  video-framework-examples build_signer
$ meson install -C build_signer
```

The `GST_PLUGIN_PATH` environment variable must be set to the local installation directory, as the signer application is implemented as a GStreamer element.

If the signer application is compiled successfully, it will be located in `local_install/bin/signer`.

To run the signer on a test file, first copy the test file into the same directory where the command is executed. In this case, the file should be placed in the parent directory:

```
$ ./local_install/bin/signer -c h264 test_h264.mp4
```

If the signer is successful, a new file with the prefix `signed_` should be generated.

## 7 Validator

The commands listed below used to compile the validator can also be found on the [AXIS Validator GitHub](#).

```
$ meson --prefix $PWD/local_install -Dvalidator=true signed-  
video-framework-examples build_validator  
$ meson install -C build_validator
```

If the validator application is compiled successfully, it will be located in `local_install/bin/validator`.

To run the validator on a test file, first copy the test file into the same directory where the command is executed. In this case, the file should be placed in the parent directory:

```
$ ./local_install/bin/validator -c h264 test_h264.mp4
```

The results can be viewed in the `validation_results` text file produced.

## 8 Product Information

Figure 1 below illustrates the product information as displayed by the validator.

```
Product Info  
-----  
Hardware ID:      931.13  
Serial Number:    B8A44F46F6B2  
Firmware version: 11.10.61  
Manufacturer:     Axis Communications AB  
Address:          www.axis.com  
-----
```

Figure 1: Product information displayed by the validator.

The validator retrieves this information from the metadata, which is stored within the MKV file, as demonstrated in Figure 2.

```

**A\q00CZJJ2**~924·µ0*F1K0Qq|QAEALWV00/·.E' 203% romm·TtG|0mmp U0--EKU'ATN=yf:AtC2000E EU-001 1+2AE |**NUAC*/<;YX<
qU\%z/IAyU6NEZñéyZ00hEN> 0zZti=1ÉiC08NEK6A# N***** |Signed Video...0A&Y+Vb0V-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEKV8qwbF8iUgy/206AQ+LKAPyNtkk g6Nrben+BqyP6l7u5f4sj6hdD7GRDrLCDXvBU
g6Nrben+BqyP6l7u5f4sj6hdD7GRDrLCDXvBUGurkU1X7F8fgMRoaV6oHQ== PUBLIC KEY
-----END PUBLIC KEY-----
G931.13 11.10.61 Firmware and Hardware ID
B8A44F46F6B2 Serial Number
www.axis.com Address
QjU65H0eÄ'~{ *1H0F!0JÄ"m.kk ) 0
hu0·IY604EEY"
ä0!

```

Figure 2: Metadata stored within the MKV file.

However, the "manufacturer" field could not be located in the MKV file when opened as a text file. To find the manufacturer, Hex Fiend was used to decode sections of the video stream as shown below.

```

. PUBLIC KEY &Q...N . BEGIN PUBLIC KEY----- MFkwEwYHKoZIzj0CAQYIKoZIzj
0DAQcDQgAEKV8qwbF8iUgy/206AQ+LKAPyNtkk g6Nrben+BqyP6l7u5f4sj6hdD7GRDrLCDXvBU
GurkU1X7F8fgMRoaV6oHQ== -----END PUBLIC KEY----- G 931.13 11.10.61 B8A4
4F46F6B2 Axis Communications AB www.axis.com !.m....9x Z.)q. %(.r .
<... t... _ . { *1H0F!0JÄ"m.kk ) . h ...I .&.4.... ..

```

Figure 3: MKV file decoded using Hex Fiend.

This indicates the "manufacturer" field is embedded within the video stream.

Furthermore, the certificates used to validate the public key were also identified using Hex Fiend. Notably, the first certificate is always unique to each specific video and its corresponding public key. This is referred to as the attestation certificate. The second certificate, known as the intermediate certificate, is included in the metadata of every signed video. A more detailed explanation of this process is provided in the following section.

```

H0F ! .J..m.k. ) . h ...I .&.4.... .. ! ...D..`!...$.Eb{.....F.~..... ---
--BEGIN CERTIFICATE----- MIIcMTCcAh6gAwIBAgIUCzUVEUJzyvyoH4BKgQkryGga6tswCgY
IKoZiZj0EAwIw ajEfMB0GA1UEChMWQXhpcyBDb21tdW5pY2F0aW9ucyBBQjEYMBYGA1UECjMPQX
hp cyBFZGdlIFZhdWx0MS0wKwYDVQDEyRBeGlzIEVkb2UgVmF1bHQgQXR0ZXN0YXRp b24gQ0Eg
RUNDIDeWHzNMjEwOTEmMTM1NjAwWhcNMzEwOTEmMTM1NjAwWjCBuTEL MAKGA1UEBhMCU0UxDTA
LBgNVBAcTBExlbmQxH2AdBgNVBAoTFkF4aXMgQ29tbXVv aWNhdGlvbnMgQUIxGDAWBgNVBAstD0
F4aXMgRWRnZSBWYXVsdDFJMEcGA1UEAxA QXhpcyBFZGdlIFZhdWx0IEF0dGVzdGF0aW9uIDA0M
DA1MDAxNTMzRkZERTJGQjQz Q0UwNDNEN0E2QTZGNzQ4MDEVMBMGA1UEBRMMQjhBNDRGNDZGNkIy
MFkwEwYHKoZI zj0CAQYIKoZIzj0DAQcDQgAEVMT3a3l1hNgE1iJkWRzzfndoNEhGfRkQkdVcCA5
z mr3u+kR7UcSR9sU1bKGDnqykGO/WLW5tFlKjnA1+P4ICaNSMFAwHQYDVRO0BBYE FNOLmG5Kf
D1D2NroI/dLmSLtQ+Q5MA4GA1UdDwEB/wQEAWIGwDAFbGgNVHSMEGDAW gBREzSmfIeeg22w1ODia
5Pg33kUZKTAKBggqhkJOPQDAGNpADBMAjEAsx4idSOD rAy8x9Y6sCGouopiBr0iWX38+499Qqa
rdie18v2NrIH19TilozBqz/vWAjEAhlcq N8ss2YhBeARr42/mRnHW11ZHy0yl76K0oYmna65Dyy
LT5y0KyBljYShwJ10v -----END CERTIFICATE----- -----BEGIN CERTIFICATE----- MII
CjjCCAE+gAwIBAgIQMALJ2gXvfCgj1rPcSRmJDAKBggqhkJOPQDDBDBcMR8w HQYDVQKQEXZBeG
lZiENvbw11bmljYXRpb25zIEFCMRGwFgYDVQLEw9BGeGlzIEVkb2UgVmF1bHQxH2AdBgNVBAMTF
kF4aXMgRWRnZSBWYXVsdCBDBQSBFQ0MwIBcNMjAx MDI2MDg1ODM0WhgPMjA1NTA2MDExMjAwMDBa
MGoxH2AdBgNVBAoTFkF4aXMgQ29t bXVuaWNhdGlvbnMgQUIxGDAWBgNVBAstD0F4aXMgRWRnZSB
WYXVsdDEtMCsGA1UE AxMkQXhpcyBFZGdlIFZhdWx0IEF0dGVzdGF0aW9uIENBIEVDQyAxMHYwEA
YHKoZI zj0CAQYFK4EEACIDYgAEligD2YyLfvMXgra74Q1K7/vam+yweU1LF5uY+qi9AthJ y7jP
V7GgIk+b/CACIinriKhOr/vUrgJuy8xN50xnC61hZvC0TkGlx1kt4m92cmFE 4Y1MgWMDbHPobyw
vxvEko2YwZDASBgNVHRMBAf8ECDAGAQH/AgEAMA4GA1UdDwEB /wQEAWIBBjAdBgNVHQ4EFgQURM
0pnyHnoNtsJtG4muT4N95FGSkwHwYDVROjBBgw FoAUkFoBqsPSSKuY+ZG8J2szqS3RcKewCgYIK
oZiZj0EAwQDgYwAMIGIAkIA9Pwn Yo+6a7KYC7tBsmPQtbcLtmRmKGR+XjHWGVkNH36JNuyjmNvU
DjVxNNbAsVy6y1g3 MLy0IC19hr2RRy1FP9QCQgCf0753SWnxgq78EPDiUg0Tn53/sintVZkUtlLU
U95y9 GZY5sFpisz7gdA17PD5tIje6eSM30Rxy2X+ufbZQm7D5w== -----END CERTIFICATE-
----- M G0E 8. P.O.. .. ;:..W. . ...s!.. ! ...}z.= .... -...E.. .V

```

Figure 4: The CA Certificates found in Hex Fiend.

## 9 Public Key Validation

### 9.1 Validation and Verification Process

The Signed Video Framework ensures the public key is validated each time before use, as specified in lines 46–48 of the [source code](#).

The framework validates the public key by verifying it against a trusted certificate chain, ensuring its authenticity through a series of signed certificates from a trusted certificate authority (CA). Specifically, the framework uses a static trusted Axis root CA certificate, which is defined on line 51 of the [source code](#).

As demonstrated in the previous section, the signed video metadata includes two certificates: the intermediate certificate, which is signed by the Axis root CA certificate. And the attestation certificate, which contains the public key used to verify the signed video, this is signed by the intermediate certificate.

The signature within the attestation certificate, which is contained in the



certificate itself, is verified using the public key of the intermediate certificate. If this verification is successful, the signature of the intermediate certificate is then verified against the Axis root CA certificate. If the intermediate certificate is valid, the root CA certificate is also verified, but since the Axis root CA is already a trusted certificate, this step is automatically considered successful. This process is detailed in the [source code](#), specifically in the function `verify_certificate_chain` at line 145.

If the attestation certificate is successfully validated, it confirms that the public key contained within the certificate has been validated by the Axis root CA certificate. The transmitted public key in the metadata is then compared to the signature in the attestation certificate, as shown in the function `verify_axis_communications_public_key` on line 372 of the [source code](#). If they match, this indicates that the transmitted public key is the same as the validated public key within the attestation certificate.

## 9.2 Ensuring Secure Public Key Validation

This piece of code, which validates the public key using certificates, seems to contradict the developers' statement:

"The framework signals the video as authentic given that the Public key can be trusted. Validating the Public key is out of scope."

However, the public key is not securely validated, It's still possible to alter the GOP signatures and replace the attestation certificate's public key with a malicious one, potentially allowing the validator to pass. Even worse, an attacker could revert the signed video to its original state, sign it again, and the validation would still succeed, with the public key appearing to be valid.

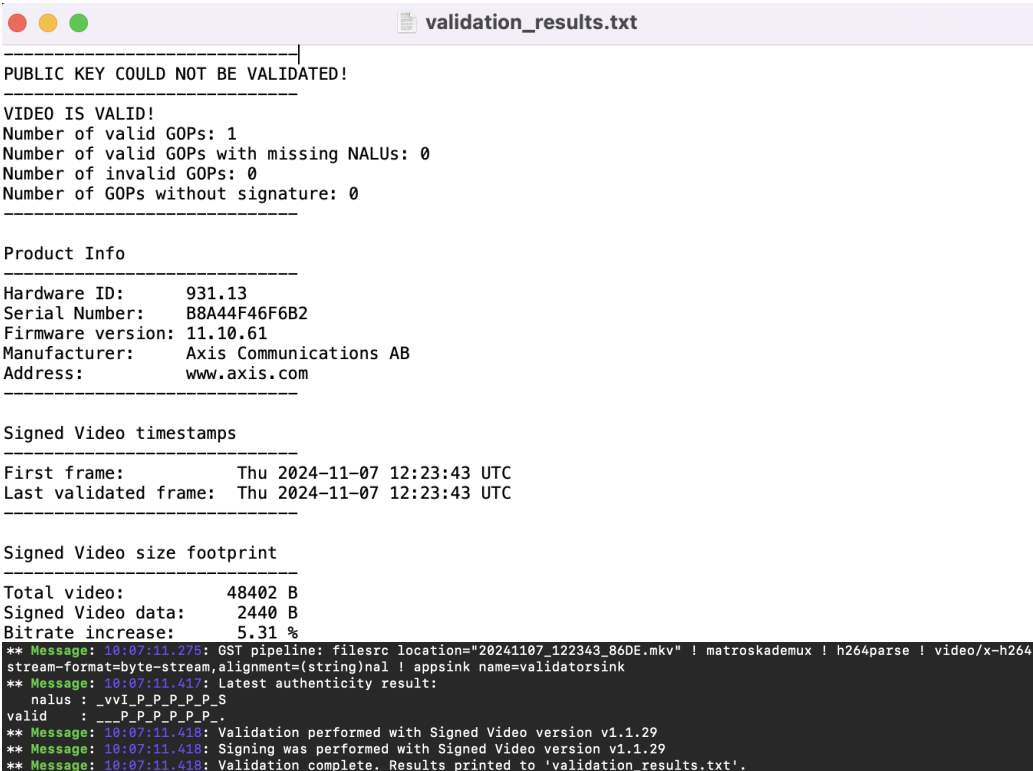
I believe a potential solution to this issue would be to store the attestation certificate in a secure storage system before distributing the signed videos. Once the video is received, the original attestation certificate can be used to validate the video's authenticity. This approach would securely ensure that the public key can be trusted when validating signed videos

## 10 Findings

### 10.1 Validating Non-Tampered Camera Footage

Figure 5 below demonstrates running the validator on a signed video captured by the AXIS Dome camera, the command used is also shown.

```
$ ./local_install/bin/validator -c h264 20241107_1223_86DE.mkv
```



```
-----
PUBLIC KEY COULD NOT BE VALIDATED!
-----
VIDEO IS VALID!
Number of valid GOPs: 1
Number of valid GOPs with missing NALUs: 0
Number of invalid GOPs: 0
Number of GOPs without signature: 0
-----

Product Info
-----
Hardware ID:      931.13
Serial Number:    B8A44F46F6B2
Firmware version: 11.10.61
Manufacturer:     Axis Communications AB
Address:          www.axis.com
-----

Signed Video timestamps
-----
First frame:      Thu 2024-11-07 12:23:43 UTC
Last validated frame: Thu 2024-11-07 12:23:43 UTC
-----

Signed Video size footprint
-----
Total video:      48402 B
Signed Video data: 2440 B
Bitrate increase: 5.31 %
** Message: 10:07:11.276: GST pipeline: filesrc location="20241107_122343_86DE.mkv" ! matroskademux ! h264parse ! video/x-h264,
stream-format=byte-stream,alignment=(string)nal ! appsink name=validatorsink
** Message: 10:07:11.417: Latest authenticity result:
  nalus : _vvi_P_P_P_P_P_S
  valid  : __P_P_P_P_P_P_
** Message: 10:07:11.418: Validation performed with Signed Video version v1.1.29
** Message: 10:07:11.418: Signing was performed with Signed Video version v1.1.29
** Message: 10:07:11.418: Validation complete. Results printed to 'validation_results.txt'.
```

Figure 5: Running the validator on a signed video from the AXIS Dome camera.

The video validation confirms two key points: the public key corresponds accurately to the private key, and the video remains unaltered, ensuring its integrity. This aligns with our expectations. However, it also shown that the public key cannot be validated. This means the validator is unable to confirm the authenticity of the public key itself.

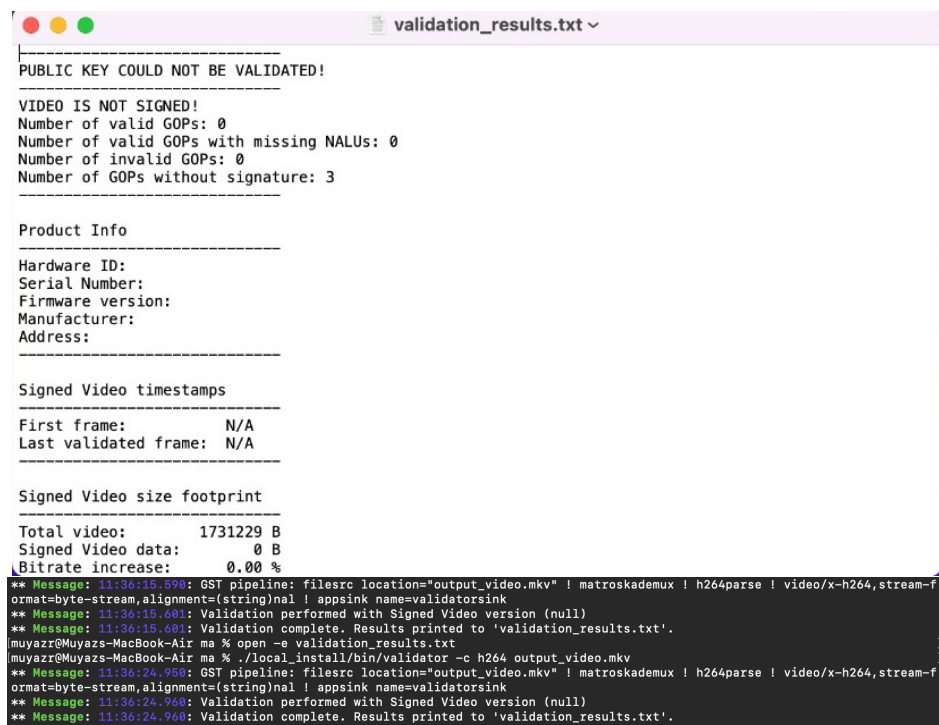
This was an unusual issue encountered while running the validator. On the Linux machine onsite, the public key was successfully validated. However, when running the validator on my personal Mac machine, the public key validation failed. I was unable to fix this issue. However, I believe it is not significant. If we can establish trust in the public key, the validator's determination that the video is authentic inherently confirms the validity of the public key.

## 10.2 Validating Tampered Camera Footage

A PNG image was inserted into the signed video from the above section. This was accomplished using `ffmpeg` with the following command:

```
$ ffmpeg -i input_video.mkv -i image.png -filter_complex
"overlay=x=10:y=10" -c:a copy output_video.mkv
```

Figure 6 below shows the result of running the validator on this altered signed video captured by the AXIS Dome camera.



```
validation_results.txt
-----
PUBLIC KEY COULD NOT BE VALIDATED!
-----
VIDEO IS NOT SIGNED!
Number of valid GOPs: 0
Number of valid GOPs with missing NALUs: 0
Number of invalid GOPs: 0
Number of GOPs without signature: 3
-----

Product Info
-----
Hardware ID:
Serial Number:
Firmware version:
Manufacturer:
Address:
-----

Signed Video timestamps
-----
First frame: N/A
Last validated frame: N/A
-----

Signed Video size footprint
-----
Total video: 1731229 B
Signed Video data: 0 B
Bitrate increase: 0.00 %
** Message: 11:36:15.590: GST pipeline: filesrc location="output_video.mkv" ! matroskademux ! h264parse ! video/x-h264,stream-f
ormat=byte-stream,alignment=(string)nal ! appsink name=validatorsink
** Message: 11:36:15.601: Validation performed with Signed Video version (null)
** Message: 11:36:15.601: Validation complete. Results printed to 'validation_results.txt'.
muyazr@Muyazs-MacBook-Air ma % open -e validation_results.txt
muyazr@Muyazs-MacBook-Air ma % ./local_install/bin/validator -c h264 output_video.mkv
** Message: 11:36:24.950: GST pipeline: filesrc location="output_video.mkv" ! matroskademux ! h264parse ! video/x-h264,stream-f
ormat=byte-stream,alignment=(string)nal ! appsink name=validatorsink
** Message: 11:36:24.960: Validation performed with Signed Video version (null)
** Message: 11:36:24.960: Validation complete. Results printed to 'validation_results.txt'.
```

Figure 6: Running the validator on a tampered signed video from the AXIS Dome camera.

Once again, the public key could not be validated. The validator correctly rejected the video, but it was not marked as invalid. Instead, the video was labeled as "not signed" rather than "invalid".

I believe this issue arises from the video being altered. In the original signed video, there were 8 signed GOPs (Group of Pictures), whereas the altered video shows only 3 unsigned GOPs. However, I don't see this as a problem because all the videos produced by the AXIS Dome camera are expected to be signed. If the validator returns "unsigned," it clearly indicates that the video has been tampered with.

A snippet of signed video from the 10 second mark to the 20 second mark was taken out, this was achieved with the following commands below:

```
$ ffmpeg -i 20241112_103959_22E6.mkv -ss 00:00:00 -to  
00:00:10 -c copy part1.mkv  
$ ffmpeg -i 20241112_103959_22E6.mkv -ss 00:00:20 -to  
00:01:00 -c copy part2.mkv  
$ echo "file 'part1.mkv'" > filelist.txt  
$ echo "file 'part2.mkv'" >> filelist.txt  
$ ffmpeg -f concat -safe 0 -i filelist.txt -c copy  
output.mkv
```

The result of running the validator is shown below.

The validator correctly marks the video as invalid. Three valid GOPs are detected, which likely correspond to the portion of the video before it was trimmed. However, after the trim, the subsequent GOPs are invalid because they depend on the previous GOPs, as demonstrated by the four invalid GOPs following the trim.

For the final test, the validator was used to assess a signed video that had been trimmed to remove the first 5 seconds, the command used to trim the video is shown below:

```
$ ffmpeg -i 20241107_122343_86DE.mkv -ss 00:00:00 -to  
00:00:05 -c copy trim.mkv
```

The result of running the validator is shown below.



```
** Message: 11:19:53.814: GST pipeline: filesrc location="part1.mkv" ! matroskademux ! h264parse ! video/x-h264,stream-format=b
yte-stream,alignment=(string)nal ! appsink name=validatorsink
** Message: 11:19:53.848: Latest authenticity result:
  nalus : _vvi_P_P_P_P_P_S
  valid  : __P_P_P_P_P_P_
** Message: 11:19:53.848: Validation performed with Signed Video version v1.1.29
** Message: 11:19:53.848: Signing was performed with Signed Video version v1.1.29
** Message: 11:19:53.848: Validation complete. Results printed to 'validation_results.txt'.
-----
validation_results.txt
-----
PUBLIC KEY COULD NOT BE VALIDATED!
-----
VIDEO IS VALID!
Number of valid GOPs: 4
Number of valid GOPs with missing NALUs: 0
Number of invalid GOPs: 0
Number of GOPs without signature: 0
-----

Product Info
-----
Hardware ID:      931.13
Serial Number:    B8A44F46F6B2
Firmware version: 11.10.61
Manufacturer:     Axis Communications AB
Address:          www.axis.com
-----

Signed Video timestamps
-----
First frame:      Tue 2024-11-12 10:39:59 UTC
Last validated frame: Tue 2024-11-12 10:40:11 UTC
-----

Signed Video size footprint
-----
Total video:      2359590 B
Signed Video data: 15896 B
Bitrate increase: 0.68 %
```

Figure 8: Running the validator on a tampered signed video from the AXIS Dome camera.

”Full tampering coverage is not guaranteed. There are still various tampering scenarios that are not handled and also, authenticity is a matter of trust.”

This may suggest that this type of tampering is not covered by the framework.