

UNIVERSITAT ROVIRA I VIRGILI

MULTIMEDIA SECURITY

---

# DRM-Protected Adaptive Video Streaming

---

IGNACIO MIGUEL RODRÍGUEZ

April 22, 2025

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Directory Structure</b>	<b>2</b>
<b>3</b>	<b>Implementation Steps</b>	<b>3</b>
<b>4</b>	<b>User Workflow</b>	<b>4</b>
4.1	Starting the System . . . . .	4
4.2	Streaming Videos . . . . .	5
4.3	Adaptive Streaming . . . . .	6
<b>5</b>	<b>Conclusion</b>	<b>6</b>
<b>6</b>	<b>Appendix: Passwords</b>	<b>6</b>

# 1 Introduction

This document describes the implementation of a DRM-protected adaptive video streaming system using MPEG-DASH. The system allows users to stream videos at different bitrates (500Kb, 1000Kb, and 2000Kb) with content protection through password-based encryption.

## 2 Directory Structure

The project is organized in the `mpegdash` directory with the following structure:

- Three main video folders (`elephant`, `popeye`, and `sintel`)
- Webpage files (`index.html`, `indexTest.html`, `generateXML.html`)
- Supporting files (`pbkdf2.js`, `server.py`, `server.pem`)
- Font file (`SourceSans Pro-Regular.otf`)

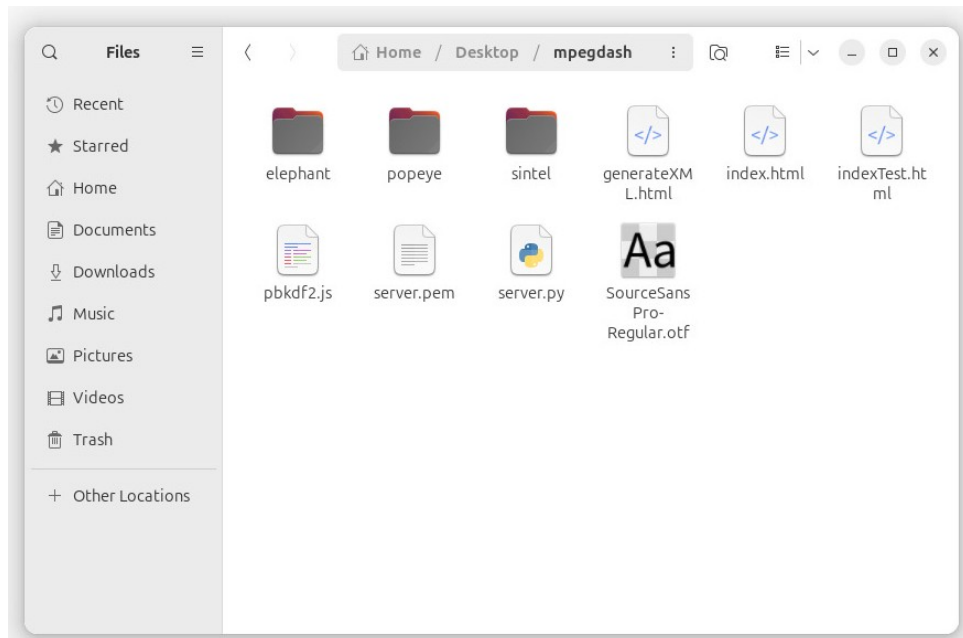


Figure 1: Directory structure overview

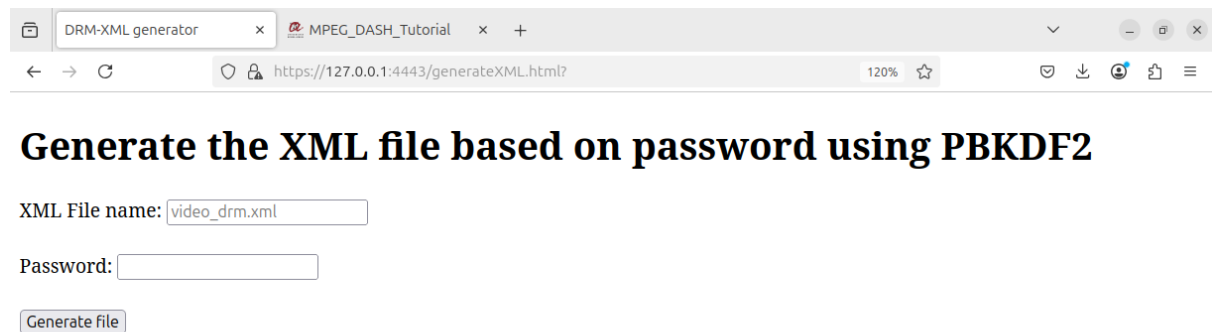
### 3 Implementation Steps

This section details the core workflow for generating encrypted video streams and their corresponding DRM manifests. The process involves two key components: (1) video encryption through adaptive bitrate encoding, and (2) dynamic XML manifest generation using password-derived cryptographic keys.

For each video (**elephant**, **popeye**, and **sintel**), the following steps were performed:

1. Created different quality streams (500Kb, 1000Kb, and 2000Kb) using **ffmpeg**.
2. Separated the audio from the video streams using **ffmpeg**.
3. Created the different DRM XML files, one per video. The implementation includes a client-side DRM configuration generator (as illustrated in Figure 2) in **generateXML.html** that creates XML manifest files following GPAC's encryption format. The system uses PBKDF2 <sup>1</sup> for password-based key derivation, generating: (1) a random 16-byte salt stored in the XML, (2) a 128-bit encryption key (1000 SHA-1 iterations), (3) a key ID (first 16 bytes of the SHA-1 of the key), and (4) a unique 8-byte IV per file. The XML structure complies with Common Encryption (CENC) standards <sup>2</sup>, containing all necessary DRM parameters while keeping cryptographic operations client-side for security. Note that after clicking on "Generate File", the XML file will be in the default *Downloads* folder, but it has to be placed inside each specific folder (e.g., *sintel*) before encrypting the streams.
4. Encrypted the streams using **MP4Box**.
5. Packaged the video and audio in DASH format (.mpd file).

Upon completing all processing steps, each video folder contains the generated files as shown in Figure 3. Due to Moodle storage limitations, non-essential files will be removed, as encouraged in the tutorial.



DRM-XML generator x MPEG\_DASH\_Tutorial x +

← → ↻ https://127.0.0.1:4443/generateXML.html? 120% ☆

## Generate the XML file based on password using PBKDF2

XML File name:

Password:

Figure 2: XML generator interface

<sup>1</sup>PBKDF2 JS's implementation: <https://gist.github.com/calvinmetcalf/91e8e84dc63c75f2aa53>

<sup>2</sup>CENC: <https://github.com/gpac/Encryption>

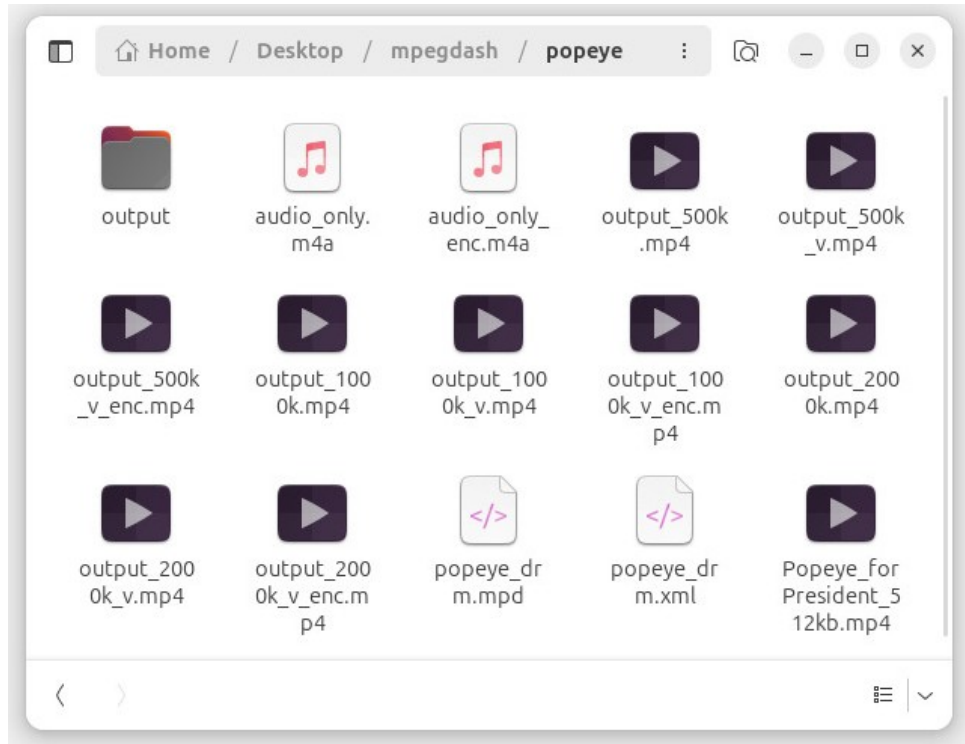


Figure 3: Example of generated video files for popeye.

## 4 User Workflow

### 4.1 Starting the System

1. Start the local server using `server.py`
2. Navigate to `https://127.0.0.1:4443` to access the interface shown in Figure 4. The page provides two options:
  - **Generate Manifest:** Used during the initial video encryption process (see Section 3).
  - **Video Streaming:** Allows playback of encrypted content by loading the manifest file and entering the associated password.



Figure 4: Main interface

## 4.2 Streaming Videos

The second option on the index page allows us to stream encrypted videos by indicating the DRM manifest and associated password from which the key and key ID should be derived.

1. Select "Video streaming" option.
2. Enter the relative path to the XML manifest file (e.g., `sintel/sintel_drm.xml`). The system requires the XML file specifically (not the `.mpd`) to retrieve the salt value, and assumes the corresponding `.mpd` file shares the same name and location.
3. Input the password originally used to generate the XML and encrypt the video streams. The player combines this password with the salt from the XML file to recreate the encryption key and key ID using PBKDF2. Note the salt is stored in the XML precisely to enable this reproducible key derivation.
4. Select your preferred bitrate from the available options.
5. Click "Load" to begin playback of the decrypted stream.

Figure 5 demonstrates the streaming interface for the `sintel` video in `indexTest.html`. Successful playback requires: (1) the correct XML manifest path (`sintel/sintel_drm.xml`) and (2) the corresponding password (listed in the document's appendix). The system's behavior varies based on input validity:

- **Incorrect password:** Infinite loading state occurs (Figure 5a) as decryption fails.
- **Correct credentials:** Video streams at selected bitrate (Figures 5b-5d).

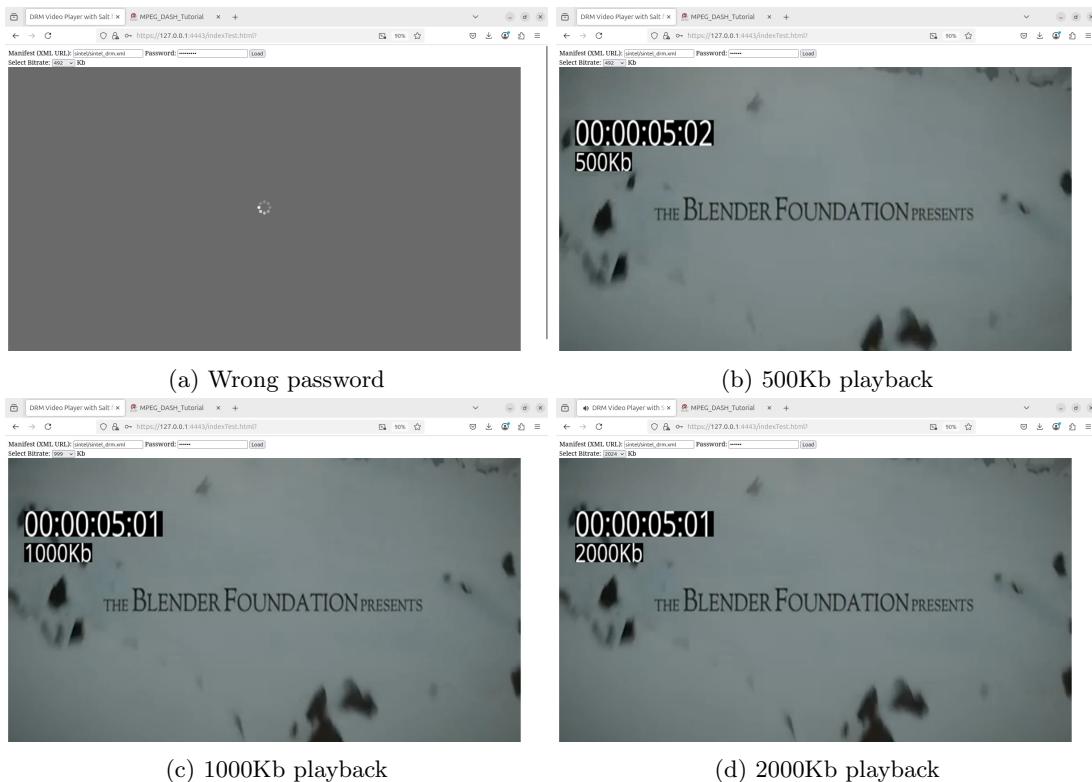


Figure 5: Video streaming interface at different stages

### 4.3 Adaptive Streaming

The system supports adaptive bitrate switching between:

- 500Kb (low quality)
- 1000Kb (medium quality)
- 2000Kb (high quality)

The actual available bitrates may vary slightly from the nominal values (e.g., 492Kb instead of 500Kb) due to encoding constraints.

## 5 Conclusion

The implemented system successfully demonstrates DRM-protected adaptive streaming using MPEG-DASH. The password-based protection ensures only authorized users can access the content, while the multiple bitrate options provide optimal viewing experience under different network conditions.

## 6 Appendix: Passwords

Table 1: Video Passwords for DRM Protection

Video Name	Password
ELEPHANT	3LePh4Nt
SINTEL	5YnT3L
POPEYE	P0p3y3