Abstract - Master's Thesis
Proof of Concept: Live Streaming C2PA-signed Content

Philip Nys

October 27, 2024

1 Motivation

Nowadays it is getting harder and harder to distinguish real, unaltered content from manipulated and automatically generated content. The tools for editing content are getting more and more accessible and easier to use and there are even many so called "GenAI" tools which are able to create more and more convincing content from a simple text prompt.

This has lead to founding of the Coalition for Content Provenance and Authenticity (C2PA) in February 2021 [1] by large players in the industry: Adobe, Arm, BBC, Intel, Microsoft and Truepic, with the goal to establish a standardized method of signing a wide range of content with metadata to verify its authenticity, origin, alterations and more.

One type of content that hasn't yet been explored in depth is live streaming fragmented media. I hereby propose a Proof of Concept (POC) for C2PA-signed live content, implement it in a testbed and analyze its effectiveness.

2 Overview C2PA

The technical specifications [2] describe the data structures, signing and verification processes, supported content types and more.

The primary component of the signature is the C2PA Manifest, which gets embedded into the content in the signing process. This Manifest consists of the Claim Signature, a digital signature of the Claim, the Claim, a manipulation-proof data structure containing all information regarding the asset, and a number of Assertions, which describe the authenticity of the asset in form of data about the creation, the steps of changes that has been made to it, metadata about the device and people having been part in making of this asset and many more.

In the current specifications there is already a wide range of formats supported, including PDF, JPEG, GIF, PNG, BMFF and more. The type of embedding varies depending on the target file, for example in case of BMFF, the relevant type for fragmented content, the Manifest is encoded into a BMFF Box "uuid" (the box is not called "c2pa" as expected, because Chromium-based browsers refuse playback of content containing unspecified BMFF boxes) and in this Box the Manifest data is structured in the JPEG universal metadata box format (JUMBF).

There is an official open source implementation of the specification in Rust in form of the *c2pa-rs* crate [3], providing an application programming interface (API). Additionally, there is a command line interface (CLI) tool, called *c2patool* [4], which provides an user interface to the API to sign content in a terminal.

[1] C2PA Founding Press Release `https://c2pa.org/post/c2pa_initial_pr/`

[2] C2PA Specification `https://c2pa.org/specifications/specifications/2.0/specs/C2PA_Specification.html`

[3] c2pa-rs `https://github.com/contentauth/c2pa-rs`

[4] c2patool `https://github.com/contentauth/c2patool`

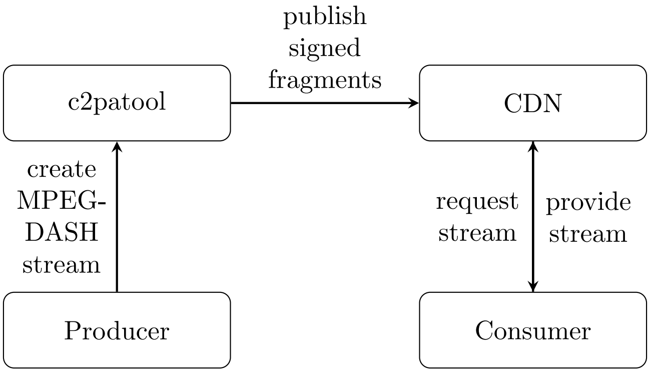

Figure 1: Proof of Concept Testbed Setup

3 Goals - Proof of Concept

The goal of this thesis is to implement a POC testbed for live streaming fragmented media and evaluate its feasibility. The proposed setup of the testbed is shown in Figure 1. The testbed will be made up of four components.

The process begins with the Producer component and will be an FFmpeg [5] script, which will generate an MPEG-DASH live stream from an input video file by endlessly loop through it. The output of this script will point to an HTTP output.

The second component will be the recipient of the HTTP output. This component will be integrated into the *c2patool* by adding a CLI flag to make the tool run as a HTTP server to receive the fragments generated by the Producer. These fragments will then be signed with a C2PA Manifest. Once the signing process is completed the signed fragments will then be forwarded to a CDN.

The Content Delivery Network (CDN) is the third component and represents a simple HTTP server which receives the signed live stream (ingest) and distributes them to clients for consumption (digest).

The final component is the Consumer which is a web-based client using *dash.js* [6] to request and play the signed stream. Additionally, the client will also verify that the received fragments are valid.

This setup will mimic a close to reality content production, where some kind of content is produced (here represented by a static video file), this then gets encoded into a fragmented live stream (here using FFmpeg). Then there is the added component of signing the content with a C2PA Manifest to prove authenticity. Ideally, these two steps would be later merged into a single step. Finally, the produced stream would be published to and hosted on a large CDN to have it available for consumption.

The big task of this setup is how to optimize the signing process. The API currently only allows to sign a complete set of fragmented BMFF files. Since a live stream is continuously generating new fragments, for every new fragment (usually roughly every two seconds) this would require the re-signing of the entire live stream up until the newly created fragment, multiplied by every available track. On top of that all fragments would also be needed to be published to the CDN anew.

4 Time Schedule

A visual representation of the following schedule is depicted in Figure 2.

I plan on having the proposal to be accepted by the end of January/early February. In the mean time I will begin with the setup of the basic testbed structure. This step will include the creation of an customizable FFmpeg script acting as the Producer, the extension of the *c2patool* by a CLI flag to make is able to receive an MPEG-DASH live stream using HTTP and forwarding fragments, the implementation of a simple HTTP setup to act as the CDN and finally a simple webpage that can play the CDN-hosted live stream using *dash.js*. I am expecting this step to take no more than four weeks.

In parallel to the initial setup I will begin writing the first draft of the thesis.

[5] FFmpeg `https://www.ffmpeg.org/`

[6] dash.js `https://github.com/Dash-Industry-Forum/dash.js`

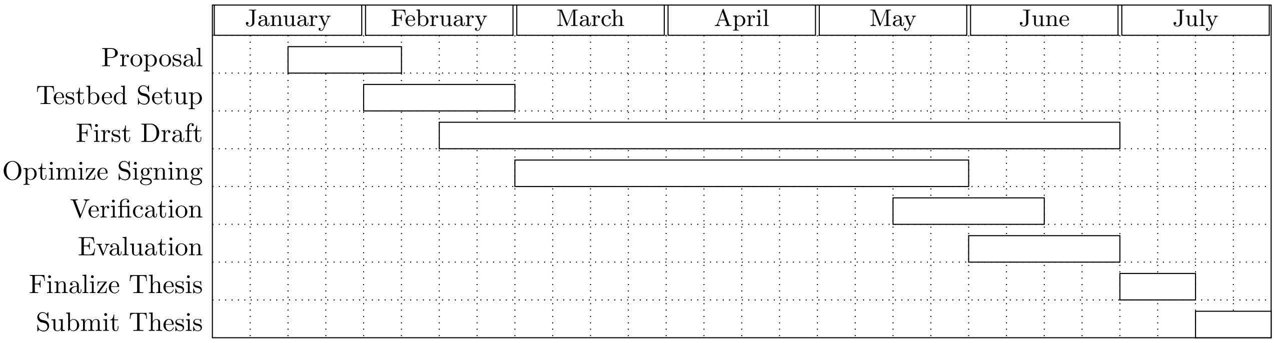

Figure 2: Time Schedule

Next, I will begin work on the signing process. Beginning with a simple test to see how this setup runs without any changes to the existing API. However, I am expecting this to become very computationally expensive the longer the live stream is running. Then I will look into optimizing this process. This step will likely be the bulk of the entire thesis and I am estimating this to take between eight and twelve weeks.

Once the live stream is being signed and can be played by the Consumer, I will implement a client-side verification of the C2PA Manifests by hooking into the event listeners of *dash.js* to inspect the received fragments, parse the C2PA ”uuid” BMMF Boxes from them and perform the verification in accordance to the specifications. I have planned between two and four weeks for this step, depending on how long the previous optimization step takes.

Similarly, I have planned the same amount of time for the evaluation to measure the performance impact of signing a live streaming before publishing and possibly more if time permits.

I aim to conclude the first draft in conjunction with the evaluation to be able to sync-up with my supervisors to discuss last changes needed to finalize the thesis to submit the results by the end of the six months at the end of July.