Update on Encrypted Media Extensions in GStreamer

Jordan Yelloz
Senior Software Engineer

About Me

- Working on GStreamer projects at Collabora since 2022
- Previously worked at Amazon Video and a few much smaller companies
  - Projects ranging from digital print automation, GStreamer, Linux audio drivers, web services
- Based in Fort Collins, Colorado, USA

- Encrypted Media Extensions (EME) Introduction/Review
- GStreamer EME Interfaces/Implementation
- GStreamer EME Shortcomings
- GStreamer EME Improvements
- Next Steps

COLLABORA

Encrypted Media Extensions

Introduction/Review

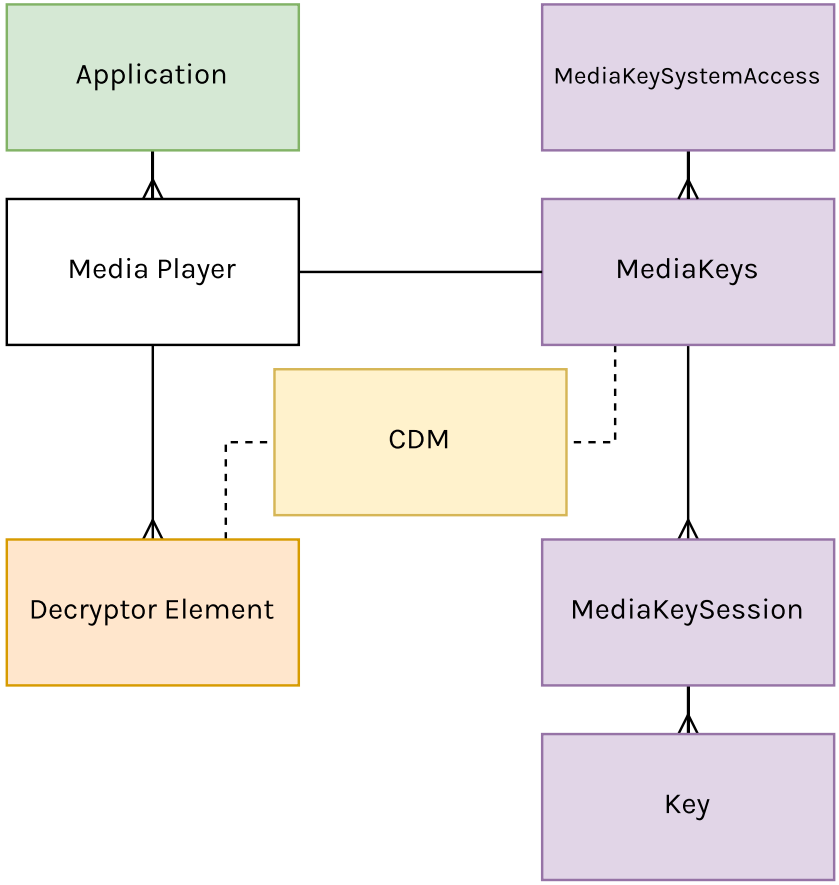
Encrypted Media Extensions - Intro

- Web technology for decryption of encrypted media
- Primarily defines communications pattern between Application, License Authority, and Content Decryption Module (CDM)
- Supported container formats:
  - MP4, WebM
- Relies on Common Encryption (CENC) scheme for each supported container
  - Allows the same encrypted media to be processed by multiple CDMs
  - Initialization Data within container informs system which keys are needed to decrypt a span of media
- Specifies "Clear Key" decryption system for evaluation purposes
- Web browsers integrate commercial CDMs

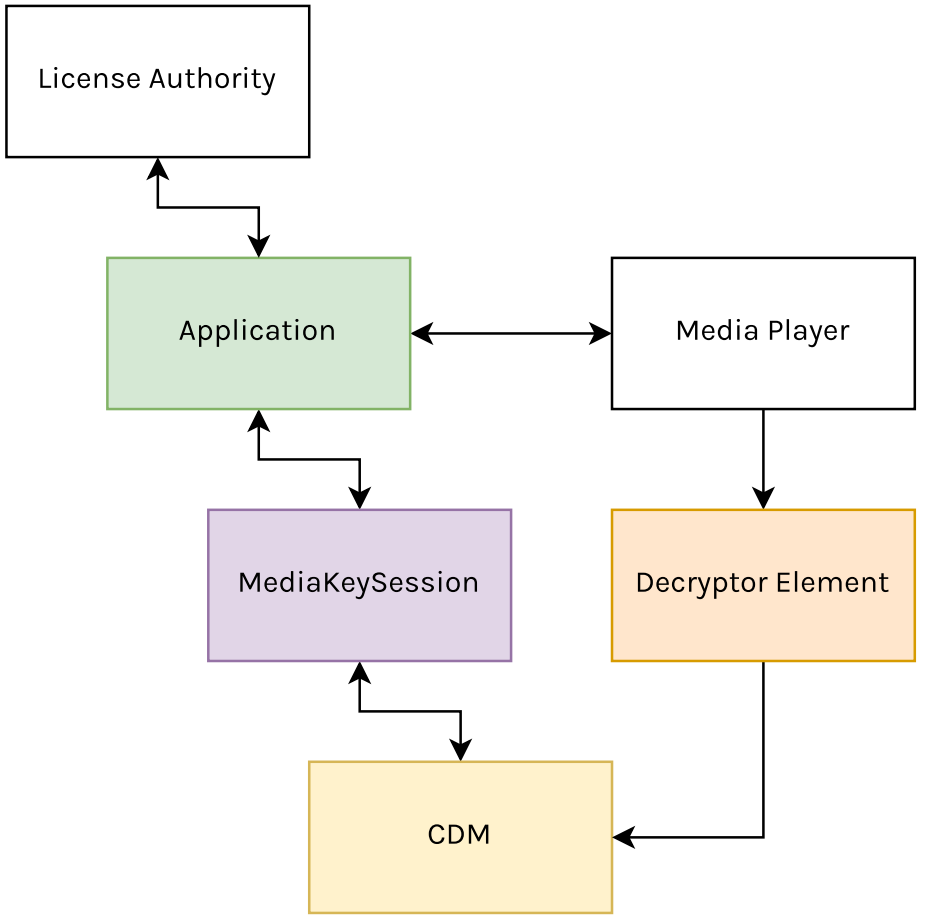
Encrypted Media Extensions - Intro

- MediaKeySystemAccess
  - Builds Media Keys instance when possible
- MediaKeys
  - Wrapper for underlying CDM instance, maintains Sessions
- MediaKeySession
  - Represents the keys referenced in a single unit of Initialization Data

EME - Structural Relationships



EME – Data Flow



Protected Media in GStreamer

- What exists now inside GStreamer?
  - Demuxers
    - Tag buffers with `GstProtectionMeta`
    - Raise `GST_EVENT_PROTECTION`
    - Supported demuxers: MP4, WebM, DASH, MSS

GStreamer EME Library

- A set of Interfaces and Data Models
  - `GstMediaKeySystemAccess` - Provides `GstMediaKeys` instance
  - `GstMediaKeys` - CDM instance wrapper, manages lifecycle of sessions
  - `GstMediaKeySession` - Groups related keys, manages lifecycle of keys
- Also defines a convention for Content Decryption Module plugins
  - Protection System - Entry point

GStreamer EME – Widevine Integration

- Using Widevine CDM included with Web Browsers
- OpenCDM module wraps Widevine library
  - Module C++ Headers are distributed in Firefox/Chrome source trees under BSD-style license
  - Discover local installation path
  - Link at runtime using GModule

GStreamer EME – Application Role

- Set up pipeline with decryptor element or just use `GstPlay`
- Instantiate supported protection system(s)
  - Request `GstMediaKeySystemAccess`
  - Create `GstMediaKeys`
- Watch the Bus for `GST_MESSAGE_NEED_CONTEXT` and inform origin element of preferred protection system
- Watch the Bus for `eme-encrypted` message from decryptor element
- Asynchronously answer contained promise with appropriate `GstMediaKeys` instance
- Create session for each new unit of Initialization Data
- Request License from License Authority
- Feed License Authority's response back to Session

COLLABORA

Encrypted Media Extensions
Updates

GStreamer EME – Shortcomings

- Complexity
- Learning Curve
- Many responsibilities for Application
  - Requires writing code to get started
  - In many situations, these can be factored out and automated

GStreamer EME – Helper Bin

- Automatic decryption when possible
  - Follows DASH Content Protection Guidelines
  - Does not require DASH input
- Customizable – Override/Supplement Parameters
  - Keys for ad-hoc decryption
  - License Authority and Authorization URLs
  - Supported/Preferred Systems
- Can work directly with gst-play or gst-launch

Development Status

- Draft Merge Request
  - https://gitlab.freedesktop.org/gstreamer/gstreamer/-/merge_requests/5640

Next Steps

- Secure Path on Android using Trusted Execution Environment

Thank you!