AAMP DASH Architecture Overview

DASH DRM Flow (analogous to MSE/EME)

1. PrivateStreamAbstractionMPD::ProcessContentProtection

- Checks if PSSH is available in manifest and, initiates DRM session creation, if PSSH is available.
- This will be the starting point for DRM session creation, if PSSH is available in manifest.
- Also Queue Protection Event using the extracted PSSH, for the cases where PSSH is missing in stream.
- This protection event is published to gstreamer layer from AMPGstPlayer_SendPendingEvents
    - **1.A)** If yes extracts the PSSH and metadata for available DRM systems
    - **1.B)** If multiple DRMs are present selects the preferred one
    - **1.C)** If lastProcessedKeyID != newKeyID

Spawns thread to create DRM session for that keyID

2. AampDRMSessionManager::createDrmSession

- CreateDrmSession, creates and/or returns the decrypt context by calling DRM API's through OpenCDMSession
- We have two slots for DRM sessions, if the new request matches none of the already processed keyIDs, clears the oldest one.
    - **2.A)** Returns drm context to decrypt if a session is present with requested keyID
    - **2.B)** Else, creates new drm session and stores the keyID to keyID cache.

AampDrmSessionFactory::GetDrmSession(systemId);

- Initializes and returns appropriate DrmSession ie, PlayReady or WideWine
- drm session state will be KEY_INIT on success

drmSession->generateAampDRMSession(initDataPtr, dataLength);

- Binds the initdata with the initialized drm session
- drm session state will be KEY_INIT on success

drmSession->aampGenerateKeyRequest(destinationURL);

- Generates license request from the drm session
- drm session state will be KEY_PENDING on success

SecClient_AcquireLicense(....); OR AampDRMSessionManager's getLicense()

- Gets the license key by posting a proper request to license server, created
- using KeyRequest, destinationURL, accessToken etc.

drmSession->aampDRMProcessKey(key);

- Key returned from AcquireLicense is fed to drm.
- drm session state will be KEY_READY on success

3. Handling GST_EVENT_PROTECTION in gst_aampcdmidecryptor_sink_event

- In normal cases, where PSSH is present in both manifest and stream AAMP will initiate to create drm session from step 1 itself and drm session will be almost ready, when qtdemux publishes the GST_EVENT_PROTECTION
- In cases where PSSH is not in stream this will be the starting point of DRM session creation
    - **3.A)** Parse the initdata in PROTECTION_EVENT and request drm session.

4. qtdemux and playready/widevine plugins.

- AAMP playready/widevine plugins are initialized with GST_RANK_PRIMARY
- This rank is changed to GST_RANK_PRIMARY + 111, on first AAMP tune.
- This is to make sure AAMP plugins are not interfering with WPE plugins

4.A) qtdemux detects stream encryption

4.B) qtdemux sends need context message for preferred-drm

4.C) bus_sync_handler in AAMPGstPlayer responds with preferred-drm

4.D) qtdemux sends protection event for preferred-drm

4.E) playready/widevine plugins receives the protection event, only the preferred plugin process it and other one discards it.

4.F) gst_aampcdmidecryptor_transform_ip is invoked, where buffer is decrypted

Below is the diagram showing MSE-EME's stack overview, we follow almost same flow in AAMP.

https://www.w3.org/TR/encrypted-media/stack_overview.svg

https://w3c.github.io/encrypted-media/format-registry/stream/mp4.html

https://w3c.github.io/encrypted-media/format-registry/initdata/cenc.html