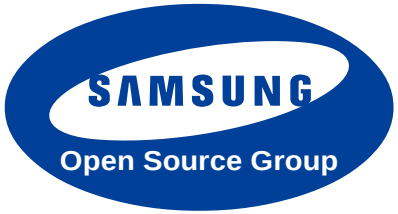
GStreamer and DRM

Thiago Sousa Santos <thiagoss@osg.samsung.com>
Luis de Bethencourt <luisbg@osg.samsung.com>
Open Source Group

SAMSUNG
Open Source Group

GStreamer

What is GStreamer?



Standard multimedia library in Linux

C / GLib / GObject

- Low-level, Multiplatform and Object Oriented

LGPL

Currently in 1.x series

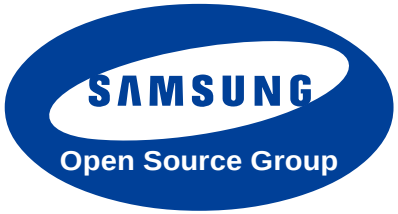
- Latest release was 1.6

15 years in development

GStreamer is everywhere

- ebook readers, Smart TVs, Conferency Systems, GPS
- Online music services, Security Cameras, ...

The problem it solves



GStreamer is modular, flexible and extensible

Functionality provided by externally shipped plugins

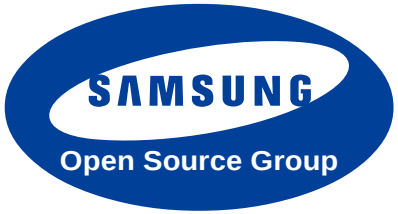
- Support binary plugins

GStreamer is a library; applications just link to it to get functionality

LGPL license allows proprietary applications

Each processing unit is treated generically and atomically

GStreamer is a pipeline



Elements are linked at connection points known as "pads"

GStreamer reaps the benefits of abstraction

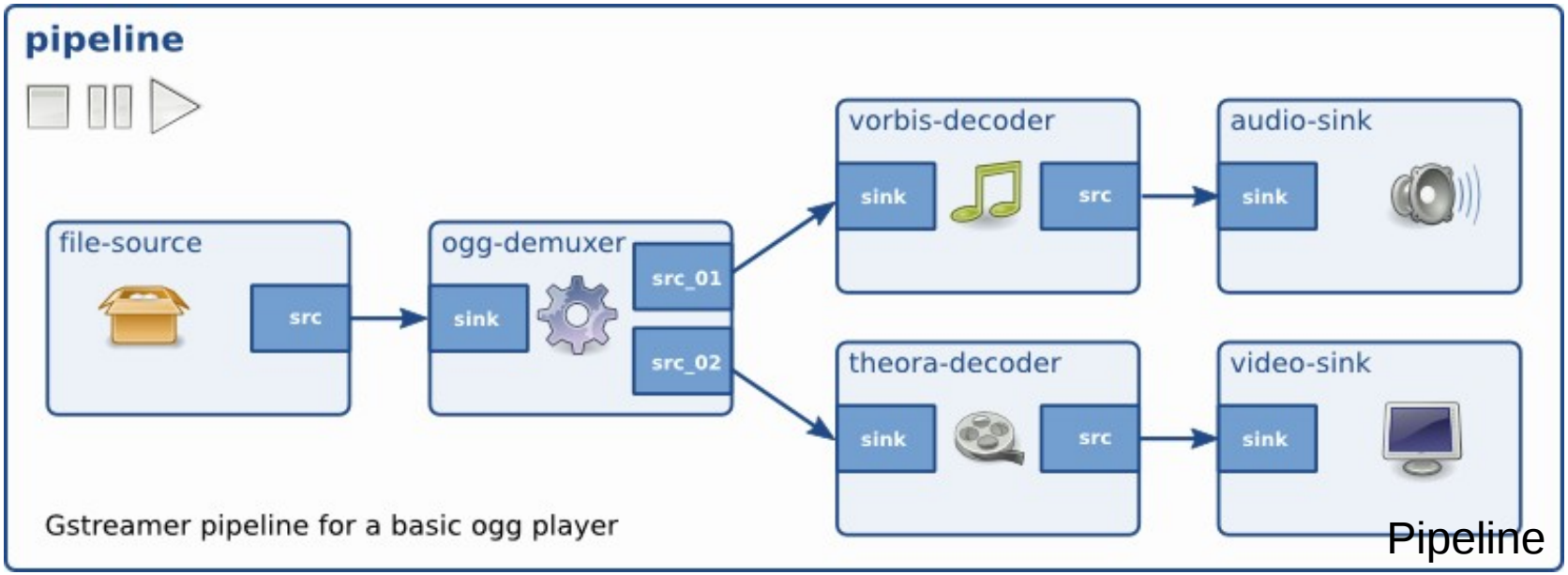
- elements + pads == arbitrary flow graph

Complex use cases are different pipeline designs

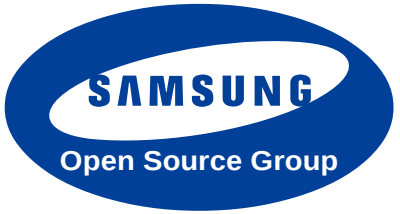
- decoding, encoding, delivery, slicing and dicing
- the elements are the same, they are reused

Unix philosophy

GStreamer is a pipeline



Two sets of APIs



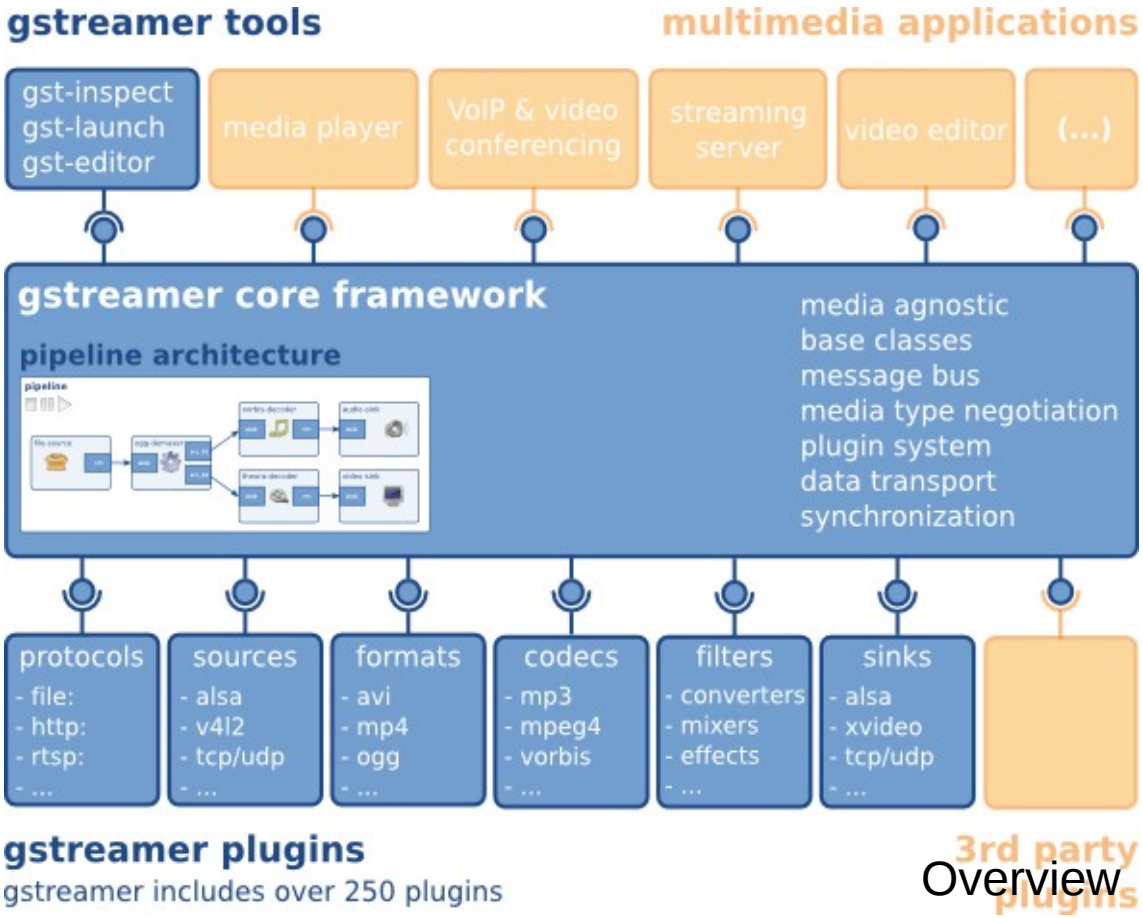
GStreamer provides two Object Oriented APIs to build multimedia systems

- API for Applications
- API for Plugins

Core architecture with small overhead

- Synchronization and clock handling
- Processing states
- Dataflow
- Events and messages
- Full multithreading
- Media agnostic
- Dynamic pipelines

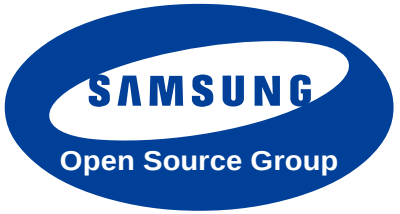
Overview



Overview

DRM

What is DRM?



DRM or Digital Rights Management is "copy protection".

A term referring to various access control technologies that are used to restrict the usage of proprietary software, hardware, or content. [0]

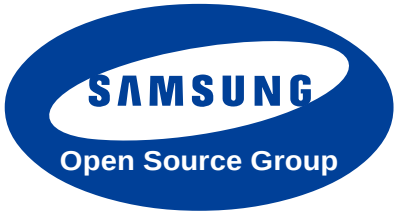
Proponents of DRM argue that it is necessary to:

- prevent intellectual property from being copied freely
- help the copyright holder maintain artistic control
- ensure continued revenue streams

The advent of digital media and analog-to-digital conversion technologies has increased the concerns of copyright-owning individuals and organizations, particularly within the music and video industries.

[0] Computer Forensics: Investigating Network Intrusions and Cyber Crime, By EC-Council

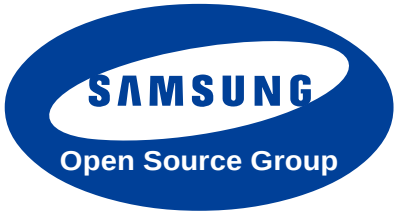
The social problem



HP and Lexmark printers, and Keurig coffeemakers

“Unfortunately, people are used to their computers telling them ‘I’m sorry you can’t do that,' But when it’s your coffee maker or your car saying ‘I’m sorry you can’t do that,’ that strikes people as very strange.” *Mitch Stoltz, Electronic Frontier Foundation*

The social problem



Very controversial topic.

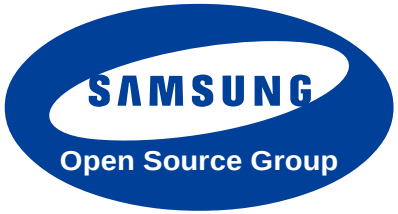
Those opposed to DRM say:

- There is no evidence that DRM helps prevent copyright infringement
- Instead, it inconveniences legitimate customers
- It helps big business stifle innovation and competition
- Works can become permanently inaccessible if the DRM scheme is discontinued
- It can also restrict users from exercising their legal rights

Criticism and opposition to PIPA (PROTECT IP Act) escalated to major websites going black for a day.

The Trans-Pacific Partnership (TPP) is a proposed multinational trade agreement. It includes points on DRM and it will compel signatory nations to enact laws banning circumvention of digital locks. Overriding each nation's copyright law, for example trumping Australia’s carefully-crafted 2007 technological protection measures regime exclusions for region-coding on movies on DVDs, video games, and players.

The technological problem

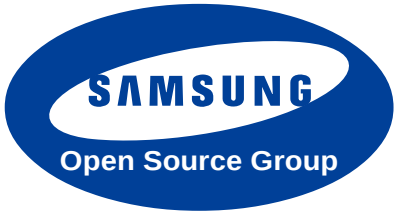


Two common tecniques are used to enforce DRM:

- **Restrictive Licensing Agreements**: usage in condition of entering a website
  - Eg: Diablo III, Starcraft 2, Always-On DRM
- **Encryption**: scrambling of material and/or embedding of a tag
  - Eg: DVD, Satellite television, EPUB books

Handling DRM in GStreamer

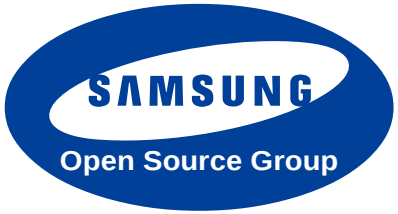
DRM in GStreamer



In release 1.6, GStreamer has gained a new generic API to signal content protection information between elements such as demuxers and decryptors. Using a very generic API, so it can support a variety of protection schemes.

The new API consists of protection events for setup information and a protection meta that can be attached to individual buffers with buffer/sample-specific information.

Content Protection signalling API



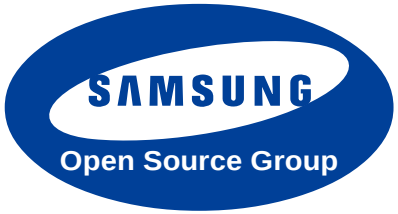
- gst_event_new_protection()
- gst_event_parse_protection()

A decryptor element needs all the protection system specific information to acquire the decryption key(s) for that stream.

The *protection event* allows elements in the pipeline to exchange that information.

- From DASH/ISOBMFF demuxers to decryptors

Content Protection signalling API

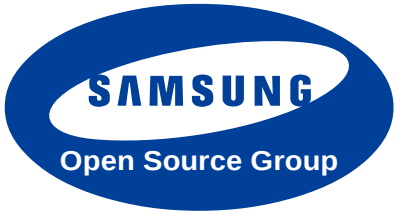



```
GstEvent *
gst_event_new_protection (const gchar *system_id,
                          GstBuffer *data,
                          const gchar *origin);
```

- **system_id**: a string holding a UUID that uniquely identifies a protection system.
- **data**: a GstBuffer holding protection system specific information.
- **origin**: a string indicating where the protection information carried in the event was extracted from.

This function creates a new event containing information specific to a particular protection system by which that protection system can acquire key(s) to decrypt a protected stream.

Content Protection signalling API



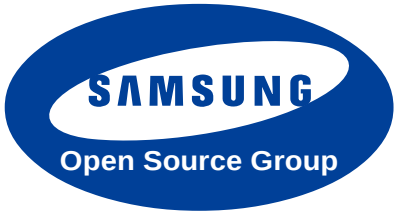
Protection systems identification and data can be in different locations. Inside the stream, on a manifest or in external resources. For **DASH CENC** it may be located:

- within ISOBMFF files, both in movie (moov) boxes and movie fragment (moof) boxes;
- in ContentProtection elements within MPEG DASH MPDs

The *protection event* contain data identifying the origin of the information.

- Some protection systems use different encodings depending upon where the information originates

Content Protection signalling API

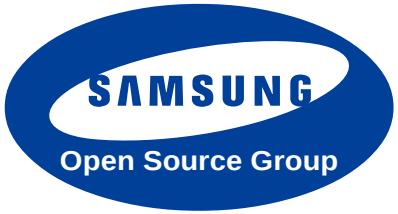



```
void
gst_event_parse_protection (GstEvent *event,
                            const gchar
**system_id,
                            GstBuffer **data,
                            const gchar
**origin);
```

- **event**: a GST_EVENT_PROTECTION event, generated by gst_event_new_protection.

Parses an event containing protection system specific information and stores the results in system_id , data and origin.

GstProtectionMeta

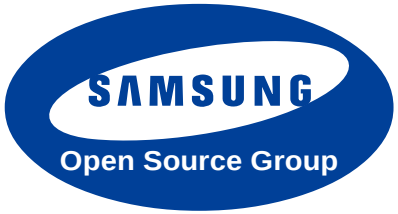


The GstProtectionMeta class enables the information needed to decrypt a GstBuffer to be attached to that buffer.

A demuxer element can attach GstProtectionMeta to the buffers that it pushes downstream.

This info can be used by a decryptor element to recover the original unencrypted frame.

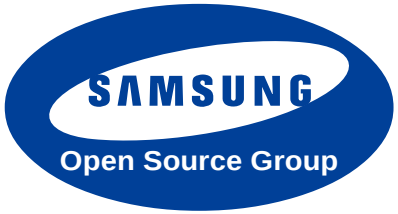
GstProtectionMeta




```
GstProtectionMeta *
gst_buffer_add_protection_meta
(GstBuffer *buffer,
 GstStructure *info);
```

- **info**: GstStructure holding cryptographic information relating to the sample contained in buffer.
- **Returns**: a pointer to the added GstProtectionMeta if successful; NULL if unsuccessful.

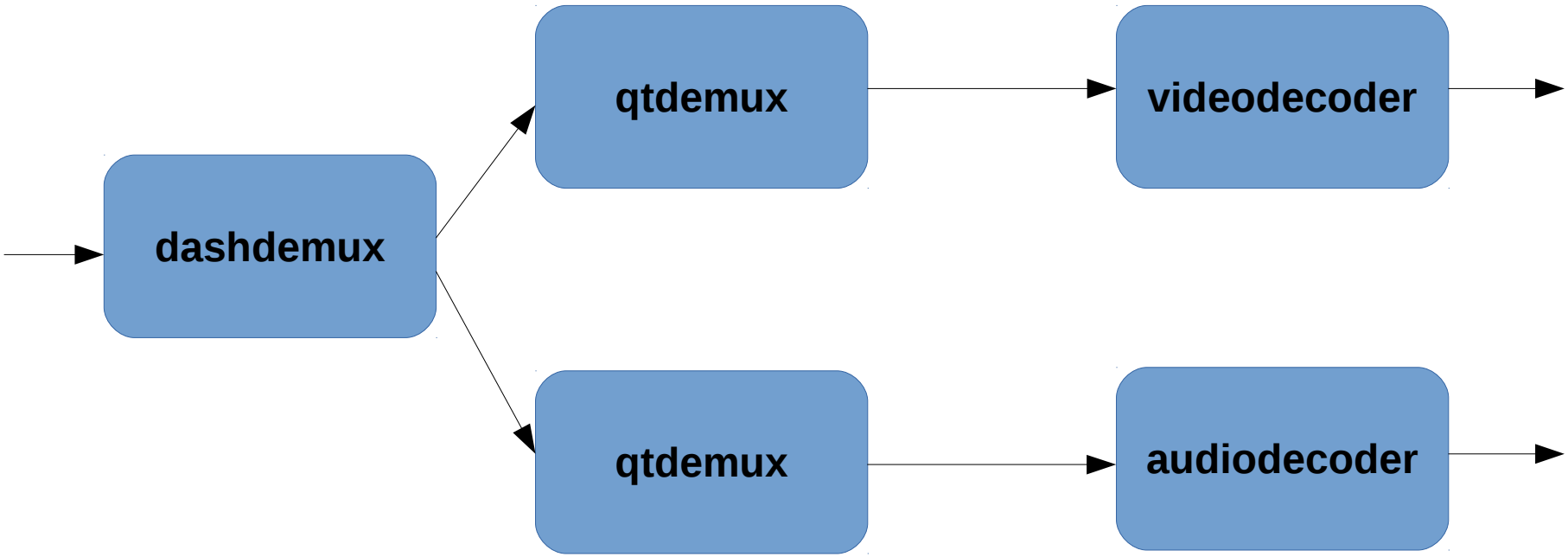
dashdemux



This new API was then used to implement support for DASH Common Encryption (CENC) in dashdemux and qtdemux. This should also cover the Common File Format (CFF) which is used e.g. by the UltraViolet format.

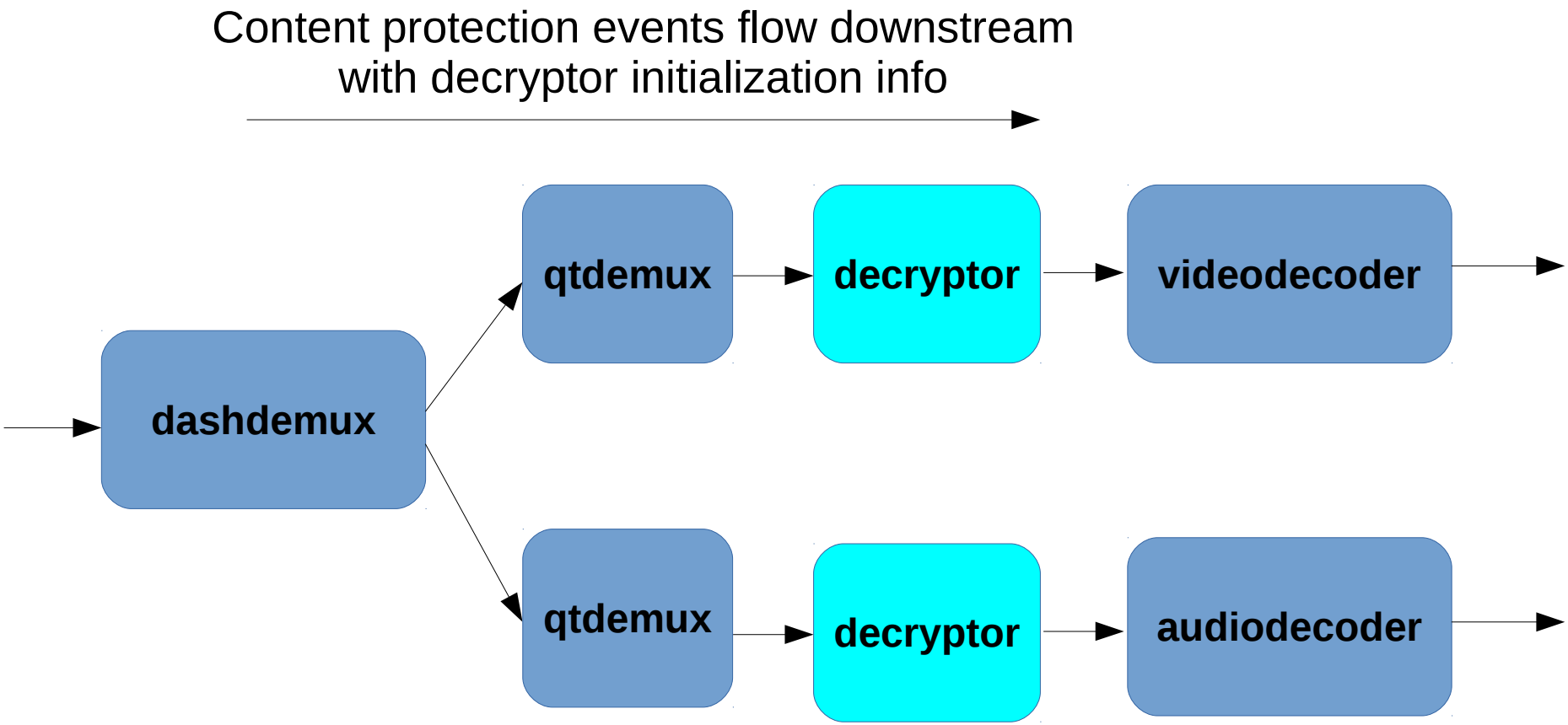
- The API is generic to be used by other DRM schemes
  - PlayReady is in bugzilla already

dashdemux



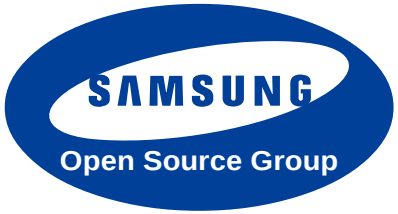
dashdemux standard playback scenario

dashdemux



dashdemux with DRM (encrypted content) playback scenario

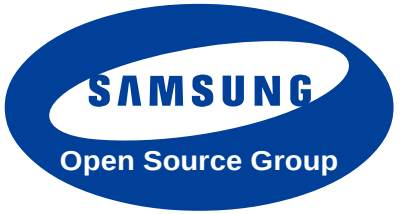
dashdemux



- Flexible approach
  - Different decryption schemes can be used
  - No changes needed on decoders
    - Transparent to them
- Can be autoplugged when needed
  - Autoplugging makes sure that decryptors are only used when necessary

Secure Memory Support

Secure Memory Support

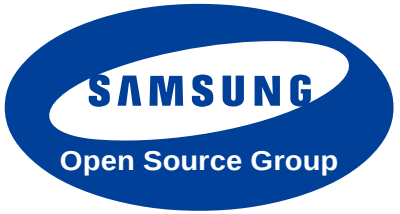


It is possible for systems to have memory that is only accessible from allowed hardware or systems.

e.g. decryptors that must decrypt to special memory that is only readable from video decoders and video rendering textures.

In such setup, it should be impossible for unallowed systems to access the decrypted data.

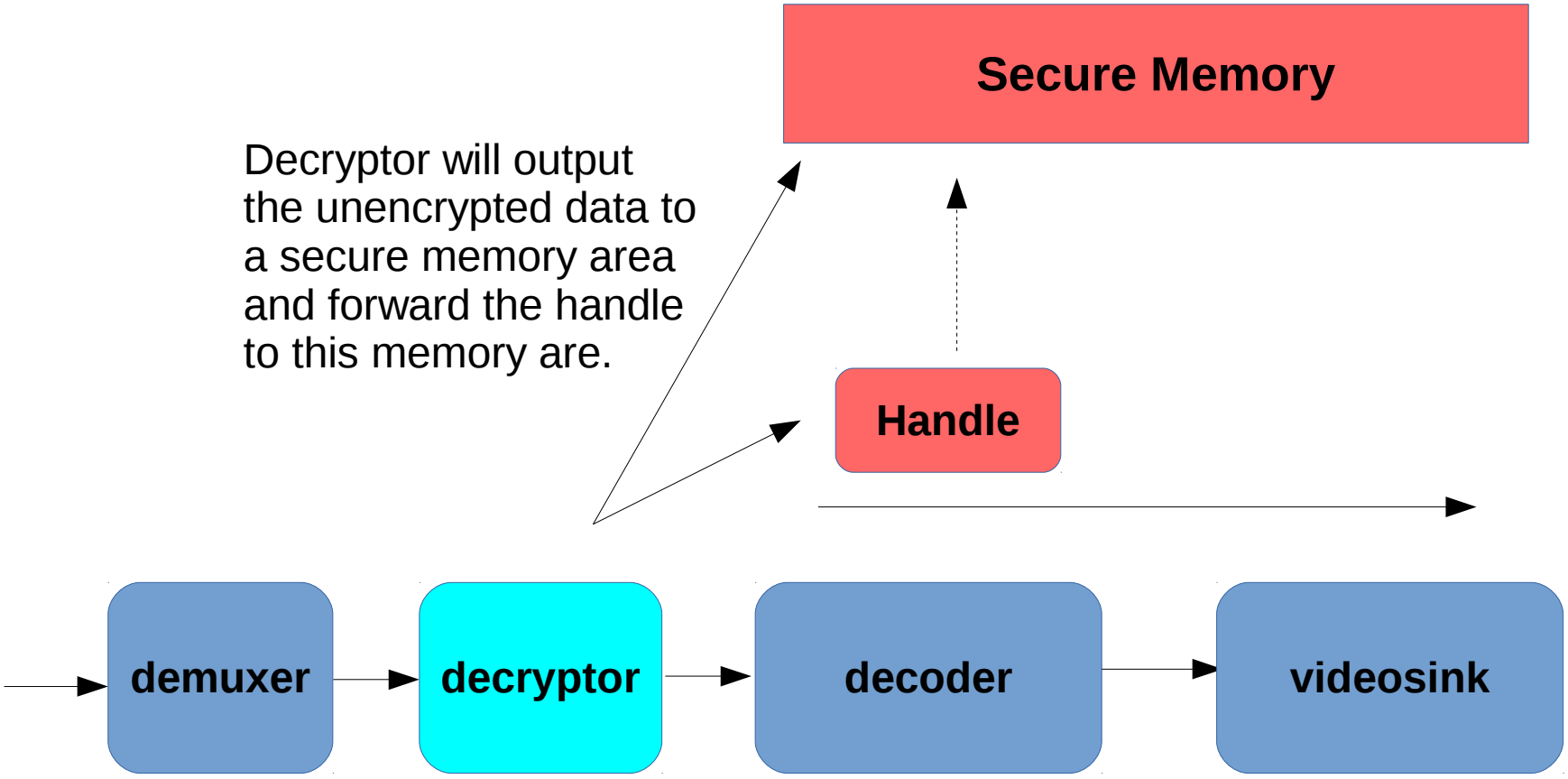
Secure Memory Support



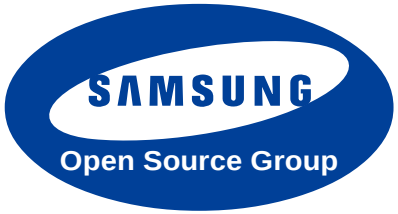
GStreamer should be able to handle secure memory buffers by holding a handle to the memory area.

- This is similar to what is done with GL data
- Elements exchange GstBuffers that contain a handle instead of the memory
  - Elements should be aware of how to operate with the handle
  - It can be announced and negotiated between elements using caps (caps features)

Secure Memory Support



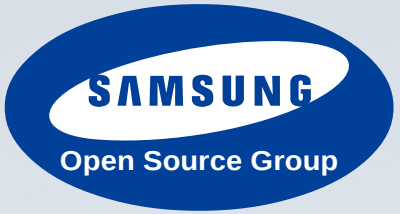
Further information



GStreamer does not ship any elements that do the actual decryption nor does it integrate with any DRM system.

That is left to device vendors and system integrators. It simply provides the infrastructure and signalling now to make it easy to add such elements.

Questions?



Thank you.