# Is DRM on the web worth it?

We've had DRM in browsers for a while now in the shape of EME and its associated technologies but we haven't asked ourselves if it really works.

To help frame the issue I've taken a section of Henri Sivonen's article What is EME to frame the post.

> **TL;DR**
>
> EME is a JavaScript API that is part of a larger system for playing DRMed content in HTML &ltvideo>/&ltaudio>. EME doesn't define the whole system. EME only specifies the JS API that implies some things about the overall system. A DRM component called a Content Decryption Module (CDM) decrypts, likely decodes and perhaps also displays the video. A JavaScript program coordinating the process uses the EME API to pass messages between the CDM and a server that provides decryption keys. EME assumes the existence of one or more CDMs on the client system but it doesn't define any or even their exact nature (e.g. software vs. hardware). That is, the interesting part is left undefined.
>
> **Context**
>
> Major Hollywood studios require that companies that license movies from them for streaming use DRM between the streaming company and the end user. Traditionally, in the Web context, this has been done by using the Microsoft PlayReady DRM component inside the Silverlight plug-in or the Adobe Access DRM component inside Flash Player. As the HTML/CSS/JS platform gains more and more capabilities, the general need to use Silverlight or Flash becomes smaller and smaller, such that soon the video DRM capability will be the only thing that Silverlight and Flash have but the HTML/CSS/JS platform doesn't.

> Proposals have been written to augment <video> with features that enable the Netflix player to be ported from Silverlight to <video> without a loss of features. The additions are split across two specifications: Media Source Extensions (MSE) and Encrypted Media Extensions (EME). The noncontroversial parts (giving JS precise control over media-related networking) are in MSE and the controversial parts (DRM interface) are in EME. I will not cover MSE further.
>
> What is EME?

With a basic understanding of what technologies are involved let's dive into EME and what all the controversy is about.

# EME and W3C

I was disappointed when I saw that EME had been published as a recommendation but I was even more disapointed that there were no provisions for researchers and archival tools excepttions. The excuse of "if we don't implement DRM it on the web companies that need/want it will go somewhere else" badly underestimates the reach of piracy and ignores that the web is not necessarily a driver for it. Encryption on the web doesn't stop people outside the web ecosystem from contributing to piracy and making the content available shortly after it becomes available.

Because the Digital Millennium Copyright Act (DMCA) (passed in 1998 in the US) and the EU Copyright Directive (passed in 2001) include provisions to prevent circumvention of DRM, it is impossible to implement DRM tools and, therefore EME support in open source products. Likewise, security researchers, people who need to modify encrypted content to enhance its accessibility, and people working to archive media content are not allowed to circumvent the EME encryption and have not access to the source material.

So yes, we can watch Netflix on the browser but, at what cost?.

# So, how does EME work?

Other than my phylosophical opposition to DRM in general, my biggest problem with EME is that it leaves a lot of behavior up to the implementors of the CDM. The theory of how EME works is relatively simple.

These steps are taken from Sam Dutton's What is EME? article in Google Developers.

1. A web application attempts to play audio or video that has one or more encrypted streams.
2. The browser recognizes that the media is encrypted and fires an encrypted event with metadata obtained from the media about the encryption.
3. The application handles the encrypted event:
    1. If no MediaKeys object has been associated with the media element, first select an available Key System to check what Key Systems are available, then create a MediaKeys object for an available Key System via a `MediaKeySystemAccess` object. **The initialization of the MediaKeys object should happen before the first encrypted event**. Getting a license server URL is done by the app independently of selecting an available key system. A `MediaKeys` object represents all the keys available to decrypt the media for an audio or video element. It represents a CDM instance and provides access to the CDM, specifically for creating key sessions, which are used to obtain keys from a license server.
    2. Once the MediaKeys object has been created, assign it to the media element, so that its keys can be used during playback, i.e. during decoding.
4. The app creates a MediaKeySession. This creates a `MediaKeySession`, which represents the lifetime of a license and its key(s).
5. The app generates a license request by passing the media data obtained in the encrypted handler to the CDM.
6. The CDM fires a message event: a request to acquire a key from a license server.
7. The MediaKeySession object receives the message event and the application sends a message to the license server (via XHR, for example).
8. The application receives a response from the license server and passes the data to the CDM
9. The CDM decrypts the media using the keys in the license. A valid key may be used, from any session within the MediaKeys associated with the media element
10. Media playback resumes.

# How does the browser know that media is encrypted?

This information is in the metadata of the media container file, which will be in a format such as ISO BMFF or WebM. For ISO BMFF this means header metadata, called the protection scheme information box. WebM uses the Matroska `ContentEncryption` element, with some WebM-specific additions. Guidelines are provided for each container in an EME-specific registry.

Note that there may be multiple messages between the CDM and the license server, and all communication in this process is opaque to the browser and application: messages are only understood by the CDM and license server, although the app layer can see what type of message the CDM is sending. The license request contains proof of the CDM's validity (and trust relationship) as well as a key to use when encrypting the content key(s) in the resulting license.

# But what do CDMs actually do?

An EME implementation does not in itself provide a way to decrypt media: it simply provides an API for a web application to interact with Content Decryption Modules.

What CDMs actually do is not defined by the EME spec, and a CDM may handle decoding (decompression) of media as well as decryption. From least to most robust, there are several potential options for CDM functionality:

- Decryption only, enabling playback using the normal media pipeline, for example via a `<video>` element.
- Decryption and decoding, passing video frames to the browser for rendering.
- Decryption and decoding, rendering directly in the hardware (for example, the GPU).

There are multiple ways to make a CDM available to a web app:

- Bundle a CDM with the browser.
- Distribute a CDM separately.
- Build a CDM into the operating system.
- Include a CDM in firmware.
- Embed a CDM in hardware.

How a CDM is made available is not defined by the EME spec, but in all cases the browser is responsible for vetting and exposing the CDM.

EME doesn't mandate a particular Key System; among current desktop and mobile browsers, Chrome supports Widevine, IE11 and Edge before its migration to Chromium support PlayReady and Safari supports FairPlay Streaming. This will become important when we look at Gatekeepers

# Is it or isn't it?

I think that the debate over EME is hanging on too much on semantics. Does it really matter how the web got its DRM do we care about the process and the way in which many people think the process was circumvented?

Adrian Rosselli makes the following point when criticizing Cory Doctorow's Boing Boing

> > [...] what we are trying to do with EME is provide a clean API to integrate these solutions with the HTML Media Element.
>
> And that's the crux of what the W3C is doing with DRM — developing a standard API so browsers can access content that will be locked down with or without their participation anyway.

This is part of the issue. While EME doesn't provide DRM directly, it still enables it and, with it, all the baggage that DRM brings with it. By leaving many aspects of the technology unspecified it makes it possible for multiple competing products to restrict content in different ways and requiring one or more licenses for the content to play in the browser at all or requiring browsers to provide incompatible solutions that are at the mercy of content producers.

One of the things that really worries me about the whole process that got EME to recommendation status was the unwillingness of the W3C leadership (and its largest members) to extend IPR protections to security researchers and people who need to break DRM to provide services to the users.

Existing copyright legislation in the US already forbids circumvention and gives copyright owners every legal justification to take you to court for doing so, even if you paid for the Blue Ray and even if you just want to make a copy so you don't

have to carry the external DVD player around.

> The moment you crack DRM (Digital Rights Managemnt) to rip the DVD, you've violated Title I of the Digital Millennium Copyright Act. 17 U.S.C. 1201 prohibits circumvention of DRM... Some courts have tried to leaven this rather harsh rule, but most have not. While it's typically hard to detect small-scale circumvention, the question is whether bypassing DRM is legal. The statute sets up some minor exceptions, but our ripper doesn't fall into any of them. So, the moment a studio protects the DVD with DRM, it gains both a technical and a legal advantage—ripping is almost certainly unlawful.
>
> Is It Legal to Rip a DVD That I Own?

If I understood this correctly, if I, as a security researcher, publish a paper or a blog post on issues with a CDM then it's up to the copyright owner and the CDM vendor to decide if they will sue me on DMCA grounds and it would be up to me to prove that I did this in good faith and with no ulterior motive.

So why would I want to open myself to this risk? Under DMCA security researchers and academics have been arrested for violating section 1201. Why wouldn't the people enabling DRM on the web want to protect me while I help hold CDM vendors accountable?

# The nature of the beast

There is no single way to implement DRM on the web and there never was. Before EME you had to decide wich plugin you wanted too use to encrypt your content. Both Silverlight and flash provided a whole development ecosystem but the plugins have been superseeded by native HTML, CSS and Javascript technologies. Now, if you decide that you want to encrypt your video, browsers need EME to play it and you need to get multiple licenses for the different encryption providers for different browsers but there's still not guaranteed that your users will be able to play the content nor that the content is fully protected.

There are further restrictions that content distributors and movie studios can "request" from CDMs vendors and EME implementors. If they don't comply then they won't get to play that encrypted content using that vendor's CDM.

Furthermore, the people who may be in the best position to ensure safety and reliability of DRM systems, security researchers can't really reverse engineer systems to work through flaws and provide a fair evaluation of what the CDM (or any DRM implementation) is doing because, unless given permission, that would be circumvention and why would DRM vendors allow people to reverse engineer their DRM products?

# What's Next?

I still have my worries as to what's the next medium to claim for their own copy protections scheme based on EME.

It comes to us as develoopers whether this is a technology that we waant to support or not or if we believe that the technology has a place in the web ecosystem.

I don't believe it does.

# Links and research

- The W3C position
  - Encrypted Media Extensions is a W3C Proposed Recommendation (Philippe Le Hégaret)
  - Encrypted Media Extensions is a W3C Proposed Recommendation
  - On EME in HTML5
- UNESCO letter
  - Encrypted Media Extensions UNESCO letter
- JustNet Coalition letter
  - Open letter from Just Net Coalition to Sir Tim Berners-Lee seeking his urgent intervention to stop acceptance of Encrypted Media Extensions as a W3C standard
- EFF and Defective by Design
  - W3C sells out the Web with EME - 1 year later
  - An open letter to the W3C Director, CEO, team and membership
- Boing Boing
  - How markets plundered Free Software's best stuff and used it to create freedom for companies, not people
  - How DRM has permitted Google to have an "open source" browser that is still under its exclusive control

- Gatekeepers and Widevine issue
    - I tried creating a web browser, and Google blocked me
    - After years of insisting that DRM in HTML wouldn't block open source implementations, Google says it won't support open source implementations
    - Google's Chrome Becomes Web 'Gatekeeper' and Rivals Complain
- The nature of the beast
    - Platform Native DRM Support
    - US Code Title 17, Section 1201
    - Expanded DMCA Exemptions Enhance Competition and Innovation
    - Exemption to Prohibition on Circumvention of Copyright Protection Systems for Access Control Technologies
    - Why can't I watch Netflix in Ultra HD on my Chrome browser?
- Is it or isn't it?
    - W3C EME is not DRM (nor other fear-mongering TLAs)
    - Requirements for DRM in HTML5 are a secret
    - Can Chromium-based Edge solve web browser's DRM issues?
    - Why can't I watch Netflix in Ultra HD on my Chrome browser?
    - Reverse Engineering of Computer Programs under the DMCA: Recognizing a "Fair Access" Defense Marquette Intellectual Property Law Review