Measuring Performance Overhead of Trans-encrypting HTTP Adaptive Streaming

Abe Wiersma BSc.

July 4, 2017

University of Amsterdam
TNO Media-lab
Problem

Major leaks of blockbuster titles.
How Hollywood Got Hacked: Studio at Center of Netflix Leak Breaks Silence (EXCLUSIVE)

Janko Roettgers
Senior Silicon Valley Correspondent
@jank0

HOLLYWOOD

Hacker Group Says ‘Hollywood Is Under Attack’ After Latest TV Leak

Tom Huston
Jan 22, 2017

The hacking collective known as The Dark Overlord is claiming that ‘Hollywood is under attack’ after the group’s latest leak of previously unreleased television episodes online earlier this week.

Broadcasters fear release of more hit shows after Dark Overlord claims leak of Orange Is The New Black

Follow Telegraph News

Follow on Facebook  Follow on Twitter  Follow on Instagram
Problem

Major leaks of blockbuster titles.

• Push to better secure DRM pipeline.
## Problem
Major leaks of blockbuster titles.

- Push to better secure DRM pipeline.

## Solution
Testing trans-encryption as an alternate form of encryption for the DRM pipeline.
Research question

- What is the performance overhead of doing a trans-encryption step for HTTP Adaptive Streaming.
  - How can available hardware efficiently be used to trans-encrypt content.
Background
HTTP Adaptive streaming

- Segment(ed/able) video.
- Manifest
- Four flavours:
  - Microsoft HTTP Smooth Streaming (HSS)
  - Adobe HTTP Dynamic Streaming (HDS)
  - Apple HTTP Live Streaming (HLS)
  - MPEG Dynamic Adaptive Streaming over HTTP (DASH)
- Traditional HTTP client/server architecture.
HTTP Adaptive streaming

Diagram showing simplified content preparation for HTTP Adaptive Streaming.
Diagram showing simplified adaptive algorithm for HTTP Adaptive Streaming.
1. Common Encryption Scheme (CENC)
   - AES-128 Cipher Block Chaining (CBC)
   - AES-128 Counter (CTR)
Digital Rights Management

Components

1. Common Encryption Scheme (CENC)
   - AES-128 Cipher Block Chaining (CBC)
   - AES-128 Counter (CTR)

2. Browser
Digital Rights Management

Components

1. Common Encryption Scheme (CENC)
   - AES-128 Cipher Block Chaining (CBC)
   - AES-128 Counter (CTR)

2. Browser

3. DRM Systems & License Servers
   - Google Widevine
   - Microsoft Playready
   - Apple Fairplay
   - Adobe Primetime
   - Others (OSS also)
Digital Rights Management

Intermission

1. Common Encryption Scheme (CENC)
   - AES-128 Cipher Block Chaining (CBC)
   - AES-128 Counter (CTR)

2. Browser
3. DRM Systems & License Servers
   - Google Widevine
   - Microsoft Playready
   - Apple Fairplay
   - Adobe Primetime
   - Other (OSS also)

Platform Native DRM Support

Please note that this table is a general guideline listing the DRM supported by popular platforms and devices. While we list the built-in DRM support that ships with each, there may be other methods of enabling additional DRM systems such as implementing an SDK. For example, our castLabs PRESTOplay SDKs provide additional DRM support for playback on iOS and Android devices.

<table>
<thead>
<tr>
<th>HTML5 Browsers</th>
<th>PlayReady</th>
<th>Widevine MODULAR</th>
<th>Widevine CLASSIC</th>
<th>FairPlay</th>
<th>Primetime (ACCESS)</th>
<th>Marlin</th>
<th>CMLA-OMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chrome (35+)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Firefox (38+ on Windows)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Firefox (47+ on Windows &amp; Mac)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internet Explorer (11+ on Windows 8.1+)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microsoft Edge (Windows 10+)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Opera (31+)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Safari (8+ on OS X)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Plugins &amp; Run-time Environments</th>
<th>PlayReady</th>
<th>Widevine MODULAR</th>
<th>Widevine CLASSIC</th>
<th>FairPlay</th>
<th>Primetime (ACCESS)</th>
<th>Marlin</th>
<th>CMLA-OMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adobe Flash / AIR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Silverlight</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mobile</th>
<th>PlayReady</th>
<th>Widevine MODULAR</th>
<th>Widevine CLASSIC</th>
<th>FairPlay</th>
<th>Primetime (ACCESS)</th>
<th>Marlin</th>
<th>CMLA-OMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android (4.3+)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Android (3+)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iOS (6+)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows Phone</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>castLabs PRESTOplay SDKs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Digital Rights Management

Components

1. Common Encryption Scheme (CENC)
   - AES-128 Cipher Block Chaining (CBC)
   - AES-128 Counter (CTR)

2. Browser

3. DRM Systems & License Servers
   - Google Widevine
   - Microsoft Playready
   - Apple Fairplay
   - Adobe Primetime
   - Others

4. Encrypted Media Extensions (EME)
Digital Rights Management

Components

1. Common Encryption Scheme (CENC)
   - AES-128 Cipher Block Chaining (CBC)
   - AES-128 Counter (CTR)

2. Browser

3. DRM Systems & License Servers
   - Google Widevine
   - Microsoft Playready
   - Apple Fairplay
   - Adobe Primetime
   - Others

4. Encrypted Media Extensions (EME)

5. Content Decryption Module (CDM)
Approach
Split-key cryptosystem

Theory

\[ D_{d_1,d_2}(E_{e_2}(E_{e_1}(X))) = X \]

- **CCU** (content consumption unit)
- **CD** (content distributor)
- **CP** (content provider)

\[ E_{e_2}(E_{e_1}(X)) \]

\[ E_{e_1}(X) \]

S
Trans-encryption\textsuperscript{1}

- RSA
- One time path
- LFSR stream cipher
- ElGamal
- Damgard-Jurik

\textsuperscript{1}As per patent: Secure distribution of content.
Split-key cryptosystem

Theory

Trans-encryption

- RSA - Widely standardized.
- One time path - Keysize increases with 100% keysize per trans-encryption.
- LFSR stream cipher - A number of insecure applications.
- ElGamal - Similar performance, hangs on discrete log, less standardized.
- Damgard-Jurik - No notable implementations.

As per patent: Secure distribution of content.
Split-key cryptosystem

RSA

\[ E(X) = X^e \pmod{n} \]
\[ D(X) = X^d \pmod{n} \]
Split-key cryptosystem

Implementation

RSA

- Generate Pair 1 (Public & Private)
- Create Pair 2 (same mod) and Combined pair (Pair 1 × Pair 2)
- Encrypt (Pair 1/Combined)
- Trans-encrypt (Encryption/Decryption 1)
- Client-decrypt (Decryption combined/Decryption 2)
Split-key cryptosystem

Implementation

RSA-2048

- `openssl genrsa`
- C `rsa_create_combined`
- Python `encrypt.py` + C `rsa_encrypt`
- C `rsa_trans/rsa_trans_dec`
- C `rsa_client_decrypt`
### Requirements

- Low overhead
- Simple
- Fast
- Free? (Opensourced)

### Solution

**Japronto**
HTTP server

Japronto!

A graph by the author squeaky-pl showing the performance of japronto.
A diagram showing the experimental set-up.
Results
Results

Throughput for HTTP Adaptive Segments

- **Passthrough MB/s**
- **AES re-encryption MB/s**
- **RSA trans-encryption (encryption) MB/s**
- **RSA trans-encryption (decryption) MB/s**

**Mean throughput MB/s** (log scale higher is better)

<table>
<thead>
<tr>
<th>Concurrent Connections</th>
<th>0MB/s</th>
<th>1MB/s</th>
<th>10MB/s</th>
<th>100MB/s</th>
<th>1000MB/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(24.51)</td>
<td>(33.69)</td>
<td>(27.76)</td>
<td>(21.73)</td>
<td>(14.5)</td>
</tr>
<tr>
<td>10</td>
<td>(111.47)</td>
<td>(75.04)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(27.76)</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td></td>
<td></td>
<td>(21.73)</td>
<td>(14.5)</td>
</tr>
</tbody>
</table>

**Required throughput for H.264 1080p streams**

1Gbit/s - Link Speed

A graph showing the throughput requesting MPEG-DASH segments applying different types of encryption.

<table>
<thead>
<tr>
<th>Throughput for HTTP Adaptive Segments</th>
<th>Passthrough MB/s</th>
<th>AES re-encryption MB/s</th>
<th>RSA trans-encryption (encryption) MB/s</th>
<th>RSA trans-encryption (decryption) MB/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean throughput MB/s (log scale higher is better)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conclusion
Conclusion

Server-side trans-encryption with the public exponent is possible

Drawback

Client-side decryption will prove tough on the performance
Future work
Future work

Possibly implement a decrypting client.
Questions?
A graph showing factorization efforts.