Measuring Performance Overhead of Trans-encrypting HTTP Adaptive Streaming

Abe Wiersma BSc.

July 4, 2017

University of Amsterdam
TNO Media-lab

Introduction

Problem

Major leaks of blockbuster titles.

Introduction

Problem

Major leaks

How Hollywood Got Hacked: Studio at Center of Netflix Leak Breaks Silence (EXCLUSIVE)

Janko Roettgers
Senior Silicon Valley Correspondent
@janko

HOLLYWOOD

Hacker Group Says 'Hollywood Is Under Attack' After Latest TV Leak

Tom Huddleston, Jr.
Jun 07, 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.

Dark Overlord strikes again: Mysterious hacker leaks new ABC TV show

Published time: 5 Jun, 2017 14:14 Get short URL

News

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

Introduction

Problem

Major leaks of blockbuster titles.

- Push to better secure DRM pipeline.

Introduction

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

Server

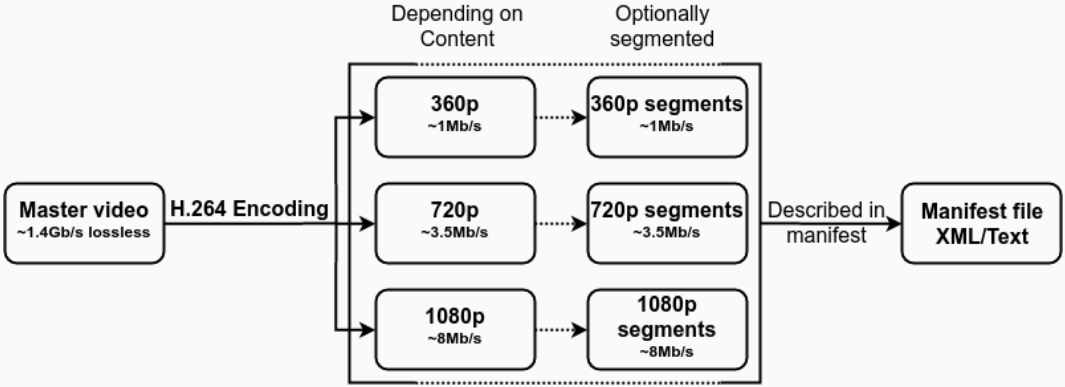


Diagram showing simplified content preparation for HTTP Adaptive Streaming.

HTTP Adaptive streaming

Client

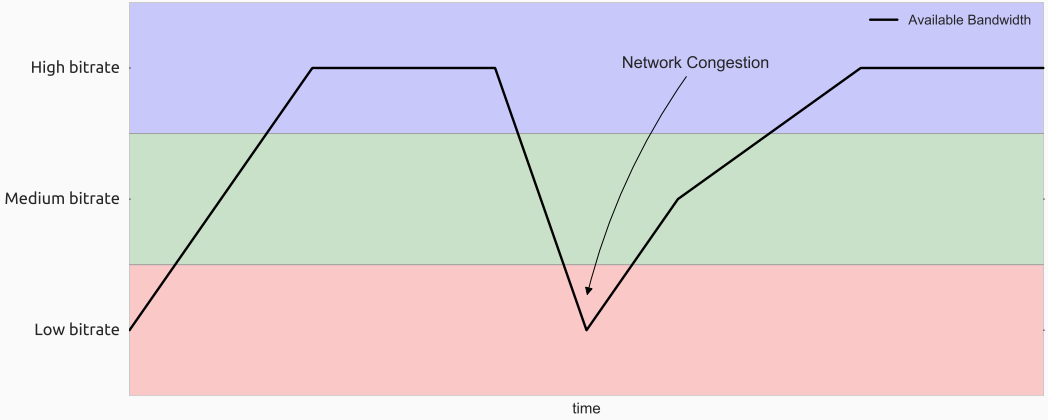


Diagram showing simplified adaptive algorithm for HTTP Adaptive Streaming.

Digital Rights Management

Components

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.

2.

3.

https://drmtoday.com/platforms/

Search

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.

HTML5 Browsers	PlayReady	Widevine MODULAR	Widevine CLASSIC	FairPlay	Primetime (ACCESS)	Marlin	CMLA-OMA
Chrome (35+)	✗	✓	✗	✗	✗	✗	✗
Firefox (38+ on Windows)	✗	✗	✗	✗	✓	✗	✗
Firefox (47+ on Windows & Mac) [1]	✗	✓	✗	✗	✓	✗	✗
Internet Explorer (11+ on Windows 8.1+)	✓	✗	✗	✗	✗	✗	✗
Microsoft Edge (Windows 10+)	✓	✗	✗	✗	✗	✗	✗
Opera (31+)	✗	✓	✗	✗	✗	✗	✗
Safari (8+ on OS X)	✗	✗	✗	✓	✗	✗	✗
Plugins & Run-time Environments	**PlayReady**	**Widevine MODULAR**	**Widevine CLASSIC**	**FairPlay**	**Primetime (ACCESS)**	**Marlin**	**CMLA-OMA**
Adobe Flash / AIR	✗	✗	✗	✗	✓	✗	✗
Silverlight	✓	✗	✗	✗	✗	✗	✗
Mobile	**PlayReady**	**Widevine MODULAR**	**Widevine CLASSIC**	**FairPlay**	**Primetime (ACCESS)**	**Marlin**	**CMLA-OMA**
Android (4.3+)	✗	✓	✓	✗	✗	✗	✗
Android (3+)	✗	✗	✓	✗	✗	✗	✗
iOS (6+)	✗	✗	✗	✓	✗	✗	✗
Windows Phone	✓	✗	✗	✗	✗	✗	✗
castLabs PRESTOplay SDKs FOR IOS & ANDROID	✗	✗	✗	✗	✗	✗	✓

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

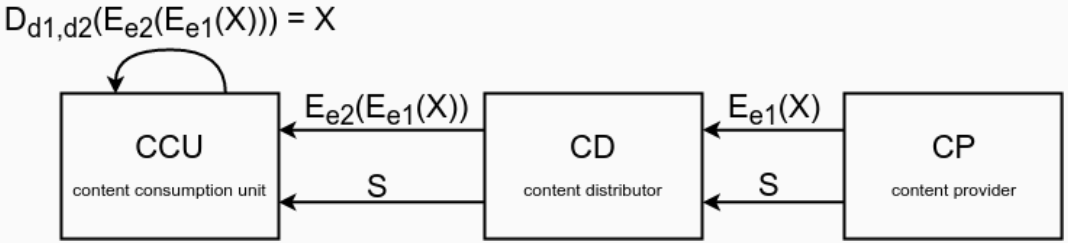
Components

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



Trans-encryption[1]

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

[1] As per patent: Secure distribution of content.

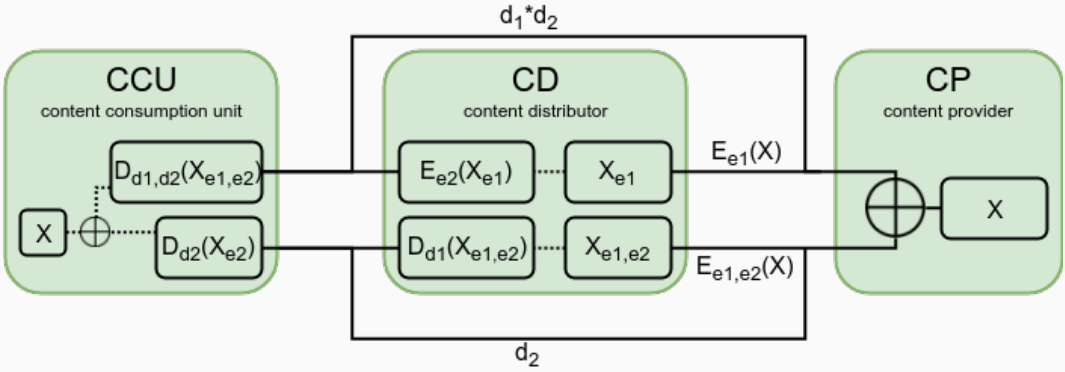
Trans-encryption[2]

- 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.

[2] As per patent: Secure distribution of content.

Split-key cryptosystem

RSA



$$E(X) = X^e \pmod{n}$$

$$D(X) = X^d \pmod{n}$$

RSA

- Generate Pair 1 (Public & Private)
- Create Pair 2 (same mod) and Combined pair (Pair 1 $\times$ Pair 2)
- Encrypt (Pair 1/Combined)
- Trans-encrypt (Encryption/Decryption 1)
- Client-decrypt (Decryption combined/Decryption 2)

RSA-2048

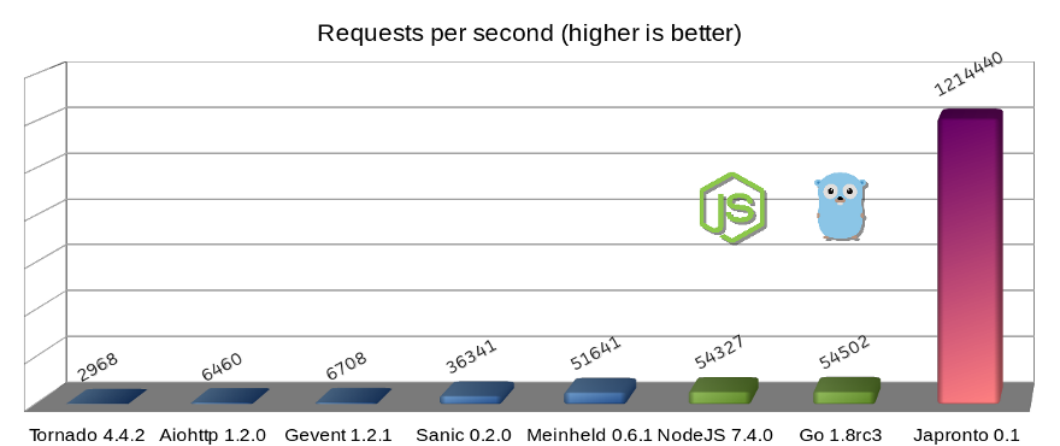
- *openssl genrsa*
- **C** *rsa_create_combined*
- **Python** *encrypt.py* + **C** *rsa_encrypt*
- **C** *rsa_trans*/*rsa_trans_dec*
- **C** *rsa_client_decrypt*

Requirements

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

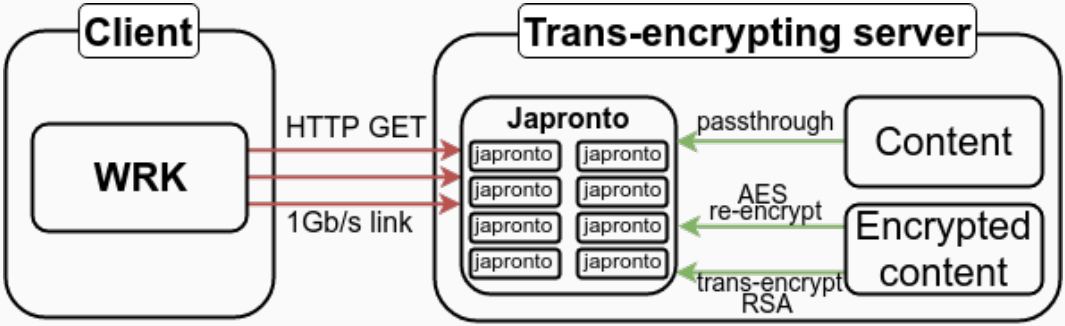
Solution

Japronto



A graph by the author *squeaky-pl* showing the performance of japronto.

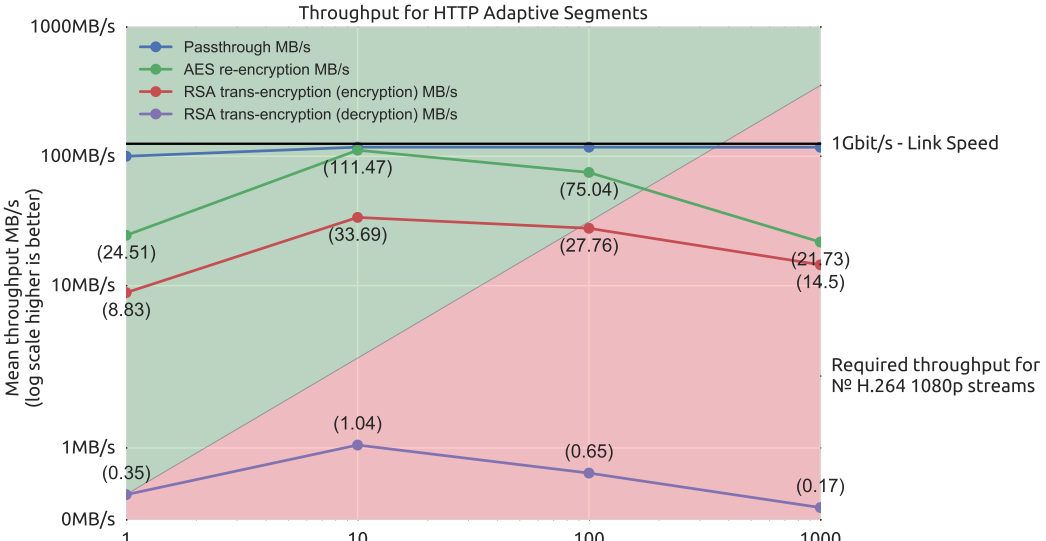
Experimental Set-up



A diagram showing the experimental set-up.

Results

Results


Throughput for HTTP Adaptive Segments
1000MB/s
100MB/s
10MB/s
1MB/s
0MB/s
Mean throughput MB/s
(log scale higher is better)
Passthrough MB/s
AES re-encryption MB/s
RSA trans-encryption (encryption) MB/s
RSA trans-encryption (decryption) MB/s
1Gbit/s - Link Speed
(111.47)
(75.04)
(24.51)
(33.69)
(27.76)
(21.73)
(14.5)
(8.83)
Required throughput for
№ H.264 1080p streams
(1.04)
(0.65)
(0.35)
(0.17)
1
10
100
1000

Conclusion

Conclusion

Conclusion

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

Drawback

Client-side decryption will prove tough on the performance

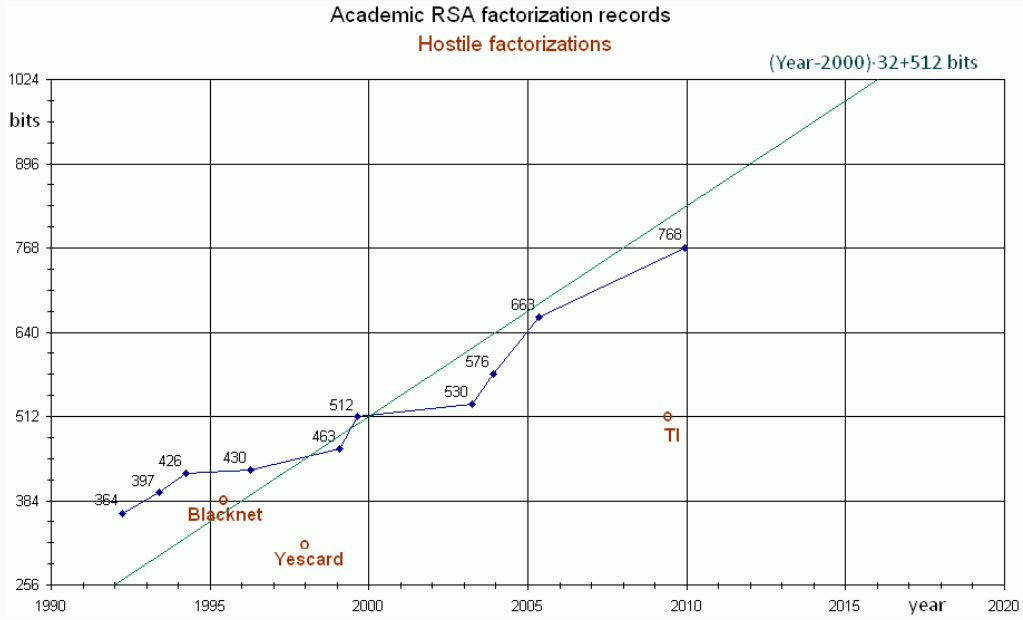
Future work

Future work

Future work

Possibly implement a decrypting client.

Questions?



A graph showing factorization efforts [3]