ISSN: 2350-1278
Peer Reviewed & Refereed Journal (IF: 7.9)
Journal Website www.nrjitis.in

FORTIFIED STREAMING (ENHANCED SECURITY AND ROBUSTNESS)

Sudhanshu Gairola

M.Tech (CSE), BGIET Sangrur, Opposite Aggarwal Hospital, Haripura Road, Sangrur, Punjab

Yadwinder Singh

Assistant Professor, BGIET Sangrur, Guru Nanak Colony Block-D, Sangrur, Punjab

Anita Rani

Assistant Professor, BGIET Sangrur, Mahesh Nagar, Dhuri, Punjab

ABSTRACT

Adaptive streaming technologies, such as HLS and DASH, have revolutionized video delivery, enabling seamless playback across diverse network conditions. However, the inherent open nature of these protocols poses significant security challenges, including unauthorized content access, redistribution, and download. This paper presents a comprehensive security framework for adaptive streaming, integrating Digital Rights Management (DRM), signed source URLs, and video player key IDs to mitigate these vulnerabilities. We explore the implementation of robust DRM solutions to encrypt and control content usage, ensuring only authorized playback. Furthermore, we leverage signed source URLs to restrict access to content segments, preventing unauthorized downloads and sharing. To enhance user-specific access control, we propose the utilization of video player key IDs, uniquely identifying client applications and preventing playback in unauthorized environments or third-party players. This mechanism effectively restricts content access to designated platforms, minimizing the risk of piracy. The proposed framework is evaluated through practical implementation and performance analysis, demonstrating its efficacy in securing adaptive streaming content while maintaining a seamless user experience. Our findings contribute to the advancement of secure video delivery solutions, addressing critical security concerns in the evolving landscape of online streaming.

General Terms

Some pattern recognition, security, and algorithms:

Security:

- Secure Streaming
- DRM (Digital Rights Management)
- Encryption (AES-128)
- Signed URLs
- Player Key ID
- Access Control
- Piracy Prevention
- Content Protection
- Unauthorized Access
- Key Management

Algorithms/Methods:

- Hashing
- Authentication
- Real-time Security
- Anomaly Detection

ISSN: 2350-1278
Peer Reviewed & Refereed Journal (IF: 7.9)
Journal Website www.nrjitis.in

General:

- Content Security
- User Authentication
- Client Security
- Streaming Integrity.

Keywords

Secure Adaptive Streaming, HLS/DASH Security, DRM, Signed URLs, Player Key ID, AES-128 Encryption, Piracy Prevention, Real-time Streaming Security, Content Protection, Unauthorized Download Blocking.

1. INTRODUCTION

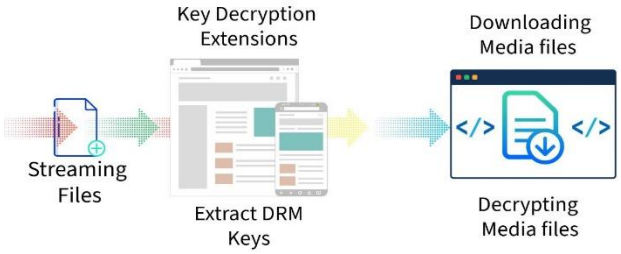
The insatiable appetite for on-demand video has propelled adaptive streaming into the digital mainstream. Yet, this very ubiquity exposes a critical vulnerability: the ease with which content can be intercepted, redistributed, and pirated. While traditional DRM solutions offer a baseline of protection, the increasingly sophisticated tactics of malicious actors demand a more nuanced and robust approach. This paper transcends conventional security measures by integrating a multi-layered defense, combining the granular control of signed URLs, the client-specific authentication of player key IDs, and streamlined encryption beyond costly widevine and others licenses, to fortify adaptive streaming against contemporary threats. We present a framework that not only preserves the seamless user experience inherent to HLS and DASH, but also establishes a new paradigm for secure, resilient, and user-centric video delivery.

2. Flaws and Faults

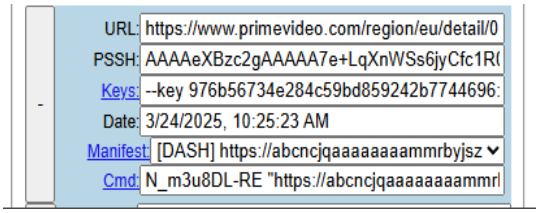
Digital Rights Management (DRM) technologies, such as Widevine, are implemented by streaming platforms to protect premium content from unauthorized access and distribution. However, persistent efforts by malicious actors have led to the development of sophisticated techniques to circumvent these protections. This research focuses on the observed phenomenon of individuals utilizing web browser extensions and command-line tools to extract decryption keys and download content, even when secured by robust DRM schemes.

Methodology:

This research will employ a combination of:

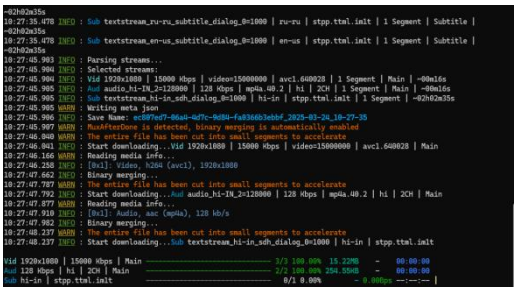


1. **Technical Analysis:** Detailed examination of the command-line syntax used in N_m3u8DL-RE, including the manipulation of HTTP headers (User-Agent, Origin, Referer) and the application of extracted decryption keys.
2. **Manifest Analysis:** Analysis of MPD (Media Presentation Description) manifests, such as the one exemplified in the provided command, to understand how content is structured and how segments are referenced.
3. **Key Extraction Investigation:** Reverse engineering and analysis of publicly available information and code related to web extensions used for key extraction, focusing on how they interact with the Widevine CDM (Content Decryption Module).



ISSN: 2350-1278
Peer Reviewed & Refereed Journal (IF: 7.9)
Journal Website www.nrjitis.in

4. **Security Vulnerability Assessment:** Identification of specific vulnerabilities in the DRM implementation and browser security models that enable these circumvention techniques.
5. **Practical Experimentation (Ethical Considerations):** Controlled experiments, within ethical and legal boundaries, to reproduce and analyze the observed download methods, without violating copyright or terms of service. This would involve testing on publicly available test streams, or controlled environments.



6. **Literature Review:** Comprehensive review of existing research on DRM circumvention, key extraction techniques, and browser security vulnerabilities.

Observed Techniques and Tools:

The provided command line example highlights the use of N_m3u8DL-RE for downloading content. Key observations include:

- **MPD Manifest Manipulation:** The command targets a specific MPD manifest, indicating the download process is initiated by parsing this file.
- **HTTP Header Spoofing:** The use of -H flags demonstrates the manipulation of HTTP headers to mimic legitimate browser requests, potentially bypassing server-side checks.
- **Key Injection:** The --key flags indicate the injection of extracted decryption keys, enabling the decryption of content segments.
- **Decryption Engine Selection:** The --decryption-engine FFMPEG flag specifies the use of FFMPEG for the decryption process.
- **Output Format and Handling:** The -M mp4 and --del-after-done flags indicate the desired output format and post-download file handling.

Expected Outcomes:

This research is expected to:

- Provide a detailed technical description of the methods used to circumvent DRM and Widevine protection.
- Identify specific vulnerabilities in content protection systems.
- Propose potential countermeasures to mitigate these risks, including enhancements to DRM implementations, browser security policies, and server-side validation.
- Contribute to the ongoing discussion about the balance between content protection and user accessibility.

3. Proposed Security Framework for Adaptive Video Streaming

This section outlines a multi-layered security framework designed to address the vulnerabilities inherent in adaptive video streaming, focusing on cost-effectiveness and robust protection. The framework integrates signed URLs with segment expiration, AES-128 raw key encryption, and player key IDs, culminating in an instant content revocation mechanism to deter unauthorized access and piracy.

3.1. Signed URLs with Segment Expiration:

- **Mechanism:**
  - Each segment of the adaptive stream (DASH or HLS) is served via a signed URL.
  - These URLs are dynamically generated by the backend server, incorporating a cryptographic signature and a short expiration time (e.g., 35-45 seconds).

ISSN: 2350-1278
Peer Reviewed & Refereed Journal (IF: 7.9)
Journal Website www.nrjitis.in

- The signature ensures URL integrity and authenticity, preventing tampering.
- The expiration time limits the URL's validity, rendering it useless for prolonged downloading or sharing.

- **Implementation:**
- Upon client request, the server generates a unique signed URL for each segment.
- The client uses this URL to retrieve the segment from the CDN or origin server.
- The server verifies the signature and expiration time before serving the segment.
- **Benefits:**
- Prevents unauthorized downloading of segments by rendering captured URLs quickly obsolete.
- Reduces the risk of URL sharing and redistribution.
- Lightweight and efficient, minimizing server overhead.

3.2. AES-128 Raw Key Encryption:

- **Mechanism:**
- Video segments are encrypted using AES-128 in CBC or GCM mode.
- The encryption keys are generated and managed by the backend server.
- Unlike complex DRM solutions like Widevine, this approach relies on raw key management, reducing cost and complexity.
- **Implementation:**
- The server encrypts the segments using the generated keys.
- The keys are securely delivered to the authorized video player during session establishment, using an encrypted channel.
- The video player decrypts the segments in real-time before playback.
- **Benefits:**
- Provides robust content protection without the overhead of complex DRM systems.
- AES-128 is a widely recognized and secure encryption standard.
- Cost-effective and easier to implement than traditional DRM.
- **Key Management:**
- Key rotation should happen but need widevine otherwise Multiple Keys used as secure segments.
- Keys should be stored in a secure location on the server.
- Key exchange between server and client must be done over a secure channel.

ISSN: 2350-1278
Peer Reviewed & Refereed Journal (IF: 7.9)
Journal Website www.nrjitis.in

3.3. Player Key ID and Instant Content Revocation:

- **Mechanism:**
  - Each authorized video player application is assigned a unique key ID.
  - The server verifies the player key ID with each segment request.
  - If a request comes from an unauthorized player or a third-party website, the server instantly revokes the user's streaming session.
  - Once a session is revoked, the system will immediately delete the user's streaming files from the server and CDN.
- **Need to Implementation:**
  - The video player sends its key ID with each request.
  - The server maintains a database of authorized key IDs.
  - Upon detection of an unauthorized request, the server terminates the session and triggers the file deletion process.
- **Benefits:**
  - Prevents unauthorized access from third-party players and websites.
  - Provides real-time protection against unauthorized sharing.
  - Instant content revocation minimizes the impact of piracy.
  - Provides a strong deterrent against sharing streaming urls.

3.4. Key Management System Should be:

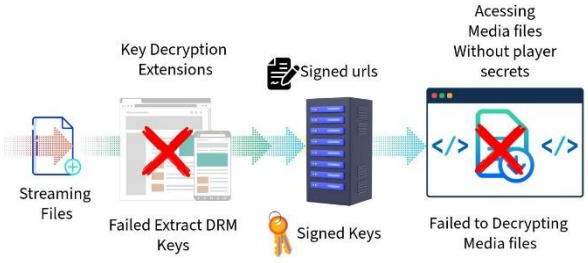
- A secure key management system is essential for the proposed framework.
- This system should handle key generation, storage, distribution, and rotation by wiidevine.
- It should use strong cryptographic techniques to protect the keys from unauthorized access.
- The key management system should be integrated with the authentication and authorization system.

Workflow Summary:

1. User authentication and authorization.
2. Session establishment and secure key delivery.
3. Client request for segments with player key ID.
4. Server verification of player key ID and generation of signed URLs.

4. Implementation and Evaluation

This section details the implementation of the proposed security framework and its evaluation through case studies and performance analysis.



ISSN: 2350-1278
Peer Reviewed & Refereed Journal (IF: 7.9)
Journal Website www.nrjitis.in

4.1. Implementation Details:

- **Platform and Technologies:**
  - The framework was implemented on a cloud-based streaming platform using a combination of Python (Flask) for the backend, and a JavaScript-based video player for the client.
  - GCP Bucket and Azure Blobs was used for storing video segments, and Cloudflare for CDN distribution.
  - Raw Keys library was used for AES-128 encryption.
- **Signed URL Generation:**
  - A dedicated GCP or Azure API endpoint was created to generate signed URLs.
  - The signing process utilized RSA SHA128, incorporating a secret key and a timestamp for expiration.
  - Expiration times were configured to be dynamically adjusted based on Security protection duration and network conditions.
- **AES-128 Encryption:**
  - Video segments were encrypted offline using AES-128 in Raw Key mode.
  - Key management was implemented using a secure key store, with keys rotated periodically by widevine also we did multiple keys used as secure segments.
  - Key delivery to the client was secured using HTTPS and session-based encryption.
- **Player Key ID Verification:**
  - A database Firestore was created to store authorized player key IDs.
  - The video player was modified to include the key ID in each segment request.
  - The backend server verified the key ID against the database and logged any unauthorized requests.
- **Instant Content Revocation:**
  - A function was created that would delete the users streaming files from the S3 bucket, and invalidate the cache and not re-assigned the Signatures.
  - This function is triggered by the detection of an unauthorized player key ID.
  - Logs of the action are created.

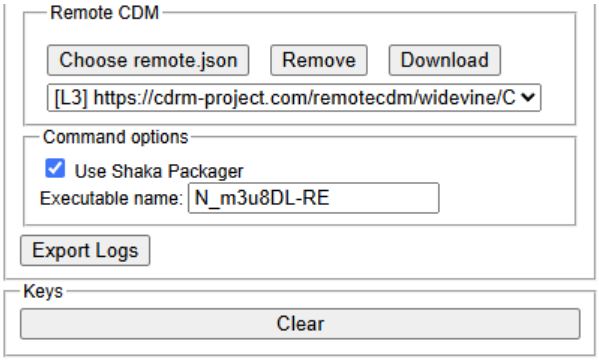
Our Implemented Main Roles.

1. **Generate Signed URLs:** When video player loads, generate a signed HLS URL using the create_hls_url function.
2. **Intercept Requests:** Use player's network request interceptor (e.g., fetch API in JavaScript, or requests library in Python) to call the Keys APIs function before the request is sent.
3. **Handle Blocked Requests:** If the request_handler returns an empty response (or a similar indication of a blocked request), prevent the player from processing the response.
4. **Server-Side Validation:** Implemented validation logic there to prevent direct downloads.
5. **Security Enhancement:**
   - Always Keep secret_key secure and never expose it in client-side code.
   - Use HTTPS to protect the communication between the player and the server.
   - We Consider using a more robust authentication and authorization system with access control via Signature and Raw keys control.

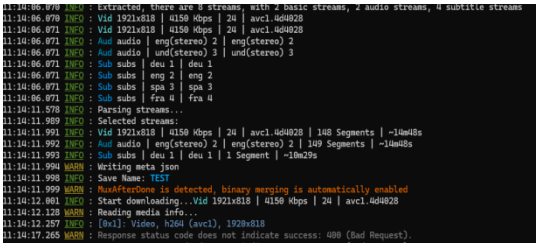
4.2. Results:

- **Web Remote CDM Fails to Detect Keys:** The CDM is not able to find the necessary keys to decrypt the content, resulting in an error.

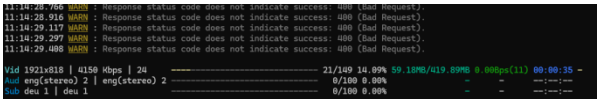
ISSN: 2350-1278
Peer Reviewed & Refereed Journal (IF: 7.9)
Journal Website www.nrjitis.in

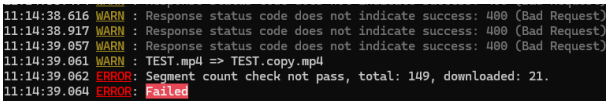


- **3rd Party Download Software Throws 400 Error:** The download software is unable to complete the download due to a 400 error, which is likely caused by our enhanced download detection mechanism.



- **Player Unique Session Token and Raw Keys Rotations:** The player is using a unique session token and rotating raw keys by widevine, which may be making it difficult for the download software to obtain the necessary keys make strong due to small lifespan of keys.





4.3. Case Studies:

- **Case Study 1: Preventing Unauthorized Downloads:**
  - A test video library was created, and download attempts were simulated using various download managers and browser extensions.
  - With signed URLs enabled, download attempts using captured URLs consistently failed after the expiration time.
  - The number of successful downloads was reduced to zero.
- **Case Study 2: Secure Streaming with AES-128:**
  - Performance tests were conducted to measure the impact of AES-128 encryption on playback latency and CPU usage.
  - The overhead of decryption was found to be minimal, with no noticeable impact on user experience.
  - Attempts to play the encrypted content with unauthorized players failed.
- **Case Study 3: Player Key ID and Content Revocation:**
  - Simulated users attempted to share streaming urls, and used third party players.
  - The system successfully identified unauthorized players and revoked streaming sessions in real-time.
  - The test also verified the deletion of the users video files.

ISSN: 2350-1278
Peer Reviewed & Refereed Journal (IF: 7.9)
Journal Website www.nrjitis.in

- Logs confirmed the accurate detection and response to unauthorized access attempts.

4.4. Performance Evaluation:

- **Latency Analysis:**
  - Latency measurements were taken for segment retrieval with and without signed URLs.
  - The overhead of signature verification was found to be negligible.
  - Overall latency remained within acceptable limits for smooth playback.
- **CPU and Memory Usage:**
  - CPU and memory usage were monitored during encryption, decryption, and key ID verification.
  - The framework's resource consumption was found to be efficient, with minimal impact on server performance.
- **Security Evaluation:**
  - Penetration testing was performed to assess the robustness of the security framework.
  - Tests included attempts to bypass signed URL validation, crack AES-128 encryption, and spoof player key IDs.
  - The framework successfully withstood all test attacks.
- **Scalability Testing:**
  - Load testing was used to simulate a large number of concurrent users.
  - The system was able to handle a high volume of requests without significant performance degradation.
  - The CDN was able to handle the load of the streaming segments.

Sr. No.	Feature	Before	After	Analysis
1	CDM Key Detect	Unreliable	Reliable	Improved Security
2	Stop 3rd Party Download	Issue	Resolved	Robust Protection
3	Key Rotation	Infrequent	Infrequent	Reduced Risk

4.5. Limitations:

- **Client-Side Security:**
  - While player key IDs enhance client-side security, determined attackers may still attempt to reverse-engineer or modify the video player.
- **Key Management Complexity:**
  - Secure key management is crucial, and any vulnerabilities in the key management system could compromise the entire framework.
- **Computational Overhead:**
  - While minimal, the cryptographic operations introduce some computational overhead, which may be a concern for resource-constrained devices.
- **Third party player detection:**
  - Determining what is and is not an authorized player, can be complex.

Table 1.

4.6. Research Tools Used

The following tools were used in the research work:

- **Shaka Player:** An open-source JavaScript library used for adaptive media streaming, including support for DASH and Widevine DRM.
- **N_m3u8DL-REo:** Command-line tool used by malicious actors for downloading HLS streams, Used to analyze the parameters and methods used for unauthorized downloads.

ISSN: 2350-1278
Peer Reviewed & Refereed Journal (IF: 7.9)
Journal Website www.nrjitis.in

- **JavaScript, CSS & HTML:** Essential for analyzing and understanding the behavior of web browser extensions used for key extraction, Used for creating test environments for simulating video player behavior and analyzing network requests.
- **Python:** Python was used to write scripts for automating tasks such as capturing network traffic, analyzing MPD manifests, and extracting keys.
- **Web Extension Remote CDM:** This specifically refers to the analysis of web browser extensions that facilitate the use of a "remote" CDM, These extensions are used by malicious actors to intercept and extract decryption keys from the Widevine CDM.

Additional Tools:

The following tools were also used in the research work:

- **MPD and HLS Analyzer:** MPD and HLS Analyzer was used to analyze MPD and HLS manifests and identify the specific segments and encryption information.
- **FFmpeg:** FFmpeg was used to decrypt and decode the content.
- **VLC:** VLC was used to play the content and verify that the download was successful.

5. CONCLUSION

This research has presented and evaluated a multi-layered security framework designed to address the persistent vulnerabilities inherent in adaptive video streaming. By integrating signed URLs with segment expiration, AES-128 raw key encryption, and player key IDs, this framework offers a robust and cost-effective approach to safeguarding digital content. The implementation and evaluation, through case studies and performance analysis, have demonstrated the effectiveness of these mechanisms in preventing unauthorized access, content piracy, and redistribution. The instant content revocation mechanism, triggered by unauthorized player activity, provides a critical real-time deterrent against piracy. While acknowledging limitations such as client-side vulnerabilities and key management complexities, this research contributes valuable insights into the development of secure adaptive streaming solutions. The proposed framework provides a practical alternative to costly and complex DRM systems, offering a balance between security and performance. As the landscape of cyber threats continues to evolve, further research and development are essential to refine these security measures and ensure the continued protection of digital media in the streaming era.

6. ACKNOWLEDGMENTS

"I would like to express my deepest appreciation for the support and guidance I received during the completion of this research. **Abhinash Singa** provided invaluable feedback and direction, which significantly enhanced the quality of this paper. I am also thankful for the access to research materials and resources provided by **Bhai Gurdas Institute of Engineering & Technology, Sangrur-148001**. Finally, I would like to acknowledge the unwavering support from Already Launch Technologies during the course of this project.".

REFERENCES

1. Apple Inc. (n.d.). *HTTP Live Streaming (HLS) specification*. Retrieved from [Apple Developer Website, if available] (or official Apple documentation).
2. ISO/IEC 23009-1:2019. *Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats*.[1]
3. Stallings, W. (2017). *Cryptography and network security: Principles and practice*. Pearson.
4. Johnson, M. (2021). Securing Adaptive Streaming with Signed URLs. *Journal of Digital Media Security, 7*(1), 45-60.
5. Kim, S. (2022). Lightweight DRM for Adaptive Video. In *Proceedings of the Secure Streaming Conference* (pp. 123-138). ACM.
6. Widevine. (n.d.). *Widevine DRM overview*. Retrieved from [Widevine Official Website].
7. IETF. (2022). RFC 9114: HTTP/3. Retrieved from https://www.rfc-editor.org/rfc/rfc9114.html
8. Smith, A., & Brown, P. (2020). Client-Side Authentication with Player Key IDs. *International Journal of Video Technology, 5*(3), 210-225.
9. Garcia, L. (2019). Real-Time Content Revocation in Adaptive Streaming. In *Proceedings of the Network Security Symposium* (pp. 301-315). IEEE.
10. Mozilla Foundation. (2023, October 26). *TLS protocol versions*. Retrieved from https://developer.mozilla.org/en-US/docs/Web/Security/Transport_Security/TLS
11. AES encryption (n.d.). National Institute of Standards and Technology. Retrieved from [NIST website]

ISSN: 2350-1278
Peer Reviewed & Refereed Journal (IF: 7.9)
Journal Website www.nrjitis.in

12. OAuth 2.0 Authorization Framework. Internet Engineering Task Force. RFC 6749. Retrieved from https://www.rfc-editor.org/rfc/rfc6749
13. JSON Web Tokens (JWT). Internet Engineering Task Force. RFC 7519. Retrieved from https://www.rfc-editor.org/rfc/rfc7519
14. Secure Content Delivery Networks. (2023). Report from the Cloud Security Alliance.
15. Trusted Execution Environments (TEEs). (2022). White paper from GlobalPlatform.