An Embedded Covert Channel Construction using HLS Protocol

Dingming Liu*, Tian Song*†, Wentian Zhao*
*School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China
†School of Education, Beijing Institute of Technology, Beijing, China

***Abstract*—The main types of modern covert channels include streaming media covert channels like VoIP covert channels, blockchain covert channels, IPv6 covert channels, TCP/IP covert channels, etc. Each method faces different challenges and limitations in application. In the field of streaming media covert channels, avoiding video content interference and reducing the difficulty of steganalysis trace detection remains a problem. Thus, this work proposes an embedded covert channel construction based on the HLS protocol. This method uses self-created TS packets and original TS packets to embed information. This method not only ensures normal video playback, but also effectively reduces the risk of covert data detection. Through experimental comparative analysis, the covert channel scheme based on the HLS protocol has advantages in terms of load capacity, transmission rate, concealment, and robustness. In particular, compared with traditional TCP/IP and VoIP covert channels, the HLS protocol can provide a more efficient and concealed covert channel, with a transmission rate of up to 500 Kbps and a stealthiness score of 0.92 against machine learning-based detection.**

***Index Terms*—HLS, covert channel, ts packet, self-created, embed.**

I. Introduction

Covert channels are communication mechanisms that allow the transmission of hidden information, typically for the purpose of bypassing security measures or ensuring privacy in scenarios where explicit communication may be monitored or restricted. These channels take advantage of unused or subtle features in data transmission protocols to encode secret messages without alerting the monitoring systems. Covert channels have been a subject of research in areas such as computer security, data privacy, and network forensics, with applications ranging from secure communication to data exfiltration.

The research and development of covert channels are of significant importance in environments where traditional communication methods are scrutinized for security reasons. As traditional techniques become increasingly detectable, more sophisticated and less conspicuous methods of embedding covert data are required. In particular, covert channels that leverage the features of existing communication protocols, like streaming protocols, offer unique advantages by making the hidden data transmission more resilient to detection and harder to differentiate from legitimate traffic.

ISBN 978-3-903176-72-0

With the rapid evolution of the Internet and the growing reliance on multimedia content for communication and entertainment, streaming protocols have become increasingly important to modern digital infrastructure. In particular, the HTTP Live Streaming (HLS) protocol, initially developed by Apple for adaptive bitrate video streaming, has become a widely used method for the distribution of live and on-demand video content. Today, HLS is extensively employed in various applications, including video-on-demand (VOD), live streaming, and short video platforms. These platforms, such as YouTube, Twitch, and TikTok, rely heavily on the scalability, adaptability, and reliability of HLS to deliver high-quality video content to users across diverse devices and network conditions.

The widespread adoption of HLS in these platforms has led to a paradigm shift in the way video content is consumed and shared. As content creators, broadcasters, and streaming services continue to push the boundaries of interactive and live content, the demand for secure, private, and sometimes covert methods of communication also grows. This opens up new opportunities for leveraging the inherent features of streaming protocols like HLS to design covert channel systems.

Traditional covert channel methods, which typically operate in lower layers of the OSI model such as the network and transport layers [1] - [3], often face challenges related to detection and security [4]. These channels can be more easily detected due to the structured and predictable nature of packet headers, flow patterns, and timing characteristics in these layers. Meanwhile, streaming protocols like HLS provide a unique advantage: their multi-layered, adaptive, and redundant structure includes a variety of fields that can be exploited for data embedding without affecting the video content's integrity. Therefore, leveraging HLS to construct covert channels represents a promising research direction.

This paper proposes a covert channel embedded within the HLS protocol by exploiting the unused stuffing bytes within the adaptation fields of Transport Stream (TS) packets. The existence of these unused stuffing bytes arises from the fixed-length mechanism of TS packets, which must always be 188 bytes in size. When the actual data payload in a packet is smaller than 188 bytes, the remaining space is filled with unused stuffing bytes to meet the fixed packet size. These

unused bytes are ideal for covert data embedding, as they do not interfere with the regular video content or affect the transmission of the streaming data. The HLS protocol itself further supports this approach due to its reliance on the Transport Stream (TS) format for segmenting video content. HLS divides video content into smaller chunks (usually 10-30 seconds long), each contained in a separate TS file. The protocol's adaptive bitrate mechanism enables efficient video streaming by dynamically adjusting the quality of the stream based on the viewer's network conditions, without disrupting the continuity of the video. This adaptability, along with the redundancy in TS packets for error correction, makes HLS an ideal candidate for embedding covert channel. By exploiting the adaptability of HLS and its widespread use in popular platforms, we can achieve a higher data transmission capacity and stealthiness compared to traditional covert channel techniques.

In this paper, we detail a systematic method for embedding and extracting hidden messages from HLS-based video streams, providing a secure covert channel within a widely-used streaming protocol. The method utilizes a pseudo-random-based transmission and reception mechanism, alongside an adaptive message segmentation algorithm. This approach ensures that the covert message is split across multiple Transport Stream (TS) packets in a manner that is both efficient and difficult to detect. Additionally, we propose a dynamic padding algorithm that further expands the capacity for embedding covert information without raising suspicion. This technique inserts padding packets at strategically calculated intervals, based on the bitrate of the stream, to increase the total available space for covert data while maintaining the integrity of the stream.

We also explore the performance of this covert channel against other common covert channel methods, demonstrating that HLS offers unique advantages in terms of data capacity and resilience to detection. This paper highlights the potential of HLS for hidden data transmission in the context of modern digital communication, offering a robust and scalable solution that aligns with the evolving needs of streaming platforms and secure communication in the digital age.

A. Contribution

To the best of our knowledge, this is the first work to design and implement a covert channel technique specifically tailored for HLS. Our key contributions are as follows:

- We introduce a new covert channel method by leveraging the adaptation field and stuffing bytes in MPEG-TS (Transport Stream) packets used in HLS streaming. This approach ensures that covert data embedding remains undetectable by standard HLS playback mechanisms.
- Beyond simply embedding covert information, we develop a complete system for both the sender and receiver to encode, transmit, and extract hidden data within the HLS protocol. This system ensures robust synchronization, efficient extraction, and reliable transmission of covert messages without disrupting the normal operation of HLS-based streaming.
- We propose a method for dynamically adjusting the padding in TS packets within HLS streams. This technique allows us to increase the covert channel's capacity based on demand while ensuring that video playback remains unaffected. This contribution enhances the flexibility and scalability of covert channel within the HLS protocol.

II. Related Work

Covert channel techniques have traditionally focused on the network and transport layers, particularly within the TCP/IP protocol stack. These techniques often involve modifying specific packet fields, leveraging timing-based methods [5], or exploiting traffic patterns to embed hidden information. Common approaches include DNS tunneling [6] and ICMP tunneling [7], which provide covert channels but may face limitations in terms of detection resistance and transmission efficiency.

With the growing importance of multimedia communication, research into audio/video-based covert channels has gained traction. One prominent area of study involves covert channels in Voice over IP (VoIP) systems, where hidden information can be embedded by manipulating packet timing [8], codec parameters [9]. Similar approaches have been explored in video streaming applications, including the modification of encoding standards (e.g., H.264, H.265) to introduce covert data within redundant frames or bitstreams [10]. While these techniques show promise, they often struggle with practical constraints such as bandwidth limitations, perceptual quality degradation, or compatibility with mainstream streaming platforms.

IPv6, as the next-generation Internet protocol, despite its design to enhance security, still presents potential attack vectors for covert channel construction. The covert channels in IPv6 [11] [12] are largely similar to those in traditional IPv4 and TCP/IP protocols, primarily exploiting padding areas in protocol fields (e.g., padding fields in the IPv6 header) and various extension headers to transmit hidden information, which means these channels also face detection and defense challenges, similar to their IPv4 counterparts [13] [14].

Blockchain technology provides a unique and inherently resilient environment for covert channel. Its decentralized nature, strong cryptographic foundations, and immutability make it difficult for third parties to alter or censor embedded information. However, the size of transaction data in most blockchain systems is strictly limited and blockchain operates with a fixed block generation interval (e.g., Bitcoin produces a new block approximately every 10 minutes) makes the blockchain-based covert channel has low payload capacity and slow transmission rate [15] [16].

Despite the widespread adoption of HTTP Live Streaming (HLS) in video-on-demand, live broadcast-

ing, and short-video platforms, covert channel within the HLS protocol has received little to no attention in existing research. To the best of our knowledge, no prior work has explicitly explored the potential of using HLS as a covert channel.

III. Framework

A. The Covert Information Embedding in the HLS Protocol

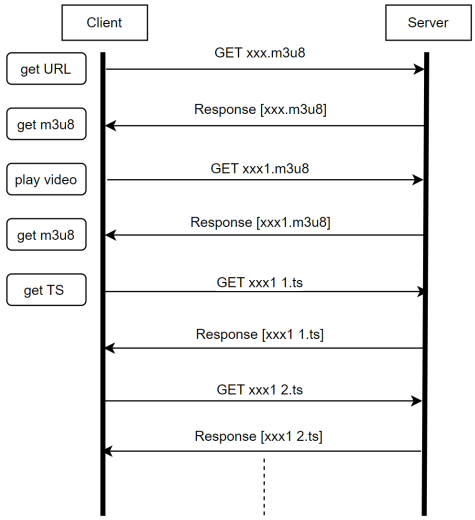

Fig. 1. HLS protocol data flow process.

As shown in Figure 1, HLS protocol is composed of three key components: HTTP, M3U8 playlists, and TS (Transport Stream) files.

Although HTTP is widely used for transporting HLS streams, it exhibits characteristics similar to traditional TCP-based covert channels when analyzed as a potential covert channel medium. Furthermore, HTTP traffic typically operates in a highly predictable manner, and information embedding would require substantial modifications to headers or payloads, making it easier to detect and interrupt covert transmissions.

M3U8 files, which are plain text files containing metadata about the streaming segments, as pure text files, M3U8 playlists have limited frequency of occurrence in the HLS streaming process, as they are generally used only for segment listing and are parsed at the start of the stream. Their limited size and plaintext nature make them inherently unsuitable for high-capacity covert data embedding. Moreover, any modification to the M3U8 file could disrupt its structure, leading to potential compatibility issues with media players. Although M3U8 files are more amenable to traditional steganographic techniques, their role in HLS is limited to segment indexing, and they are not an ideal location for embedding large amounts of covert information.

In contrast to HTTP and M3U8 files, TS (Transport Stream) files are particularly suitable for covert information embedding. A key feature of TS files is that they consist of small, fixed-size packets (188 bytes per packet) called TS packet. This predictability allows for greater flexibility when modifying the contents of these packets without disrupting the overall stream. Furthermore, TS packets can carry multiple types of data, including media content, time-stamps, and optional metadata, making them more flexible in terms of embedding additional information.

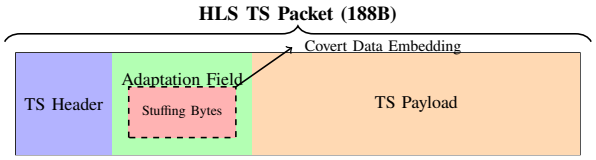

Fig. 2. Structure of an HLS TS Packet and Steganographic Data Embedding Area

As shown in Figure 2, the presence of the adaptation field is determined by the "adaptation_field_control" byte in the TS packet header. When the value indicates the presence of an adaptation field (i.e., values 2 or 3), the packet includes additional space after the 4-byte header for carrying supplementary data. This space is referred to as "stuffing bytes" within the adaptation field. Importantly, while not every TS packet contains an adaptation field, those that do offer sufficient capacity for covert data storage. In our statistical analysis of an 8.53MB TS video file, we found that the stuffing bytes within the adaptation fields of the TS packets amounted to 29,586 bytes. This amount of space is more than enough for embedding significant amounts of covert data without disrupting the primary video content. This characteristic makes the TS packet structure and its adaptation field an ideal location for embedding hidden information within HLS streams. By leveraging this space efficiently, we can design covert channel methods that exploit the existing protocol without compromising the integrity or playback of the video content.

In comparison to embedding covert data directly within the video stream, embedding data in the adaptation_field's stuffing_bytes within the HLS protocol offers several distinct advantages, especially in terms of embedding capacity and stealthiness. While video steganography techniques, such as modifying the least significant bits (LSB) of pixel values or using more advanced methods like DCT-based or matrix embedding, can provide an effective means of hiding data within the video content, they are inherently limited by the available space within the video frames and the potential for perceptible distortions.

Directly embedding covert information into video frames can indeed offer high bandwidth, but this comes at the cost of video quality. As the embedded data increases, even state-of-the-art techniques like motion compensation or matrix embedding may result in visual or auditory distortions, which can be detected through simple analysis techniques or steganalysis tools. This makes video steganography vulnerable to detection, especially when large amounts of data are

embedded or when the video content needs to be transmitted over extended periods.

In contrast, the stuffing_bytes in the adaptation_field do not alter the video content. These fields are designed to carry filler data without affecting the video or audio, maintaining the integrity of the video stream and offering a higher level of stealthiness. Unlike video steganography, which can introduce visible or audible artifacts, using the adaptation_field ensures that the embedded data remains undetected through standard streaming processes.

Moreover, since adaptation_field and stuffing_bytes are integral parts of the MPEG-TS protocol, their presence does not raise suspicion, further reducing detection risks. Although embedding data directly into the video stream may provide higher capacity, it increases the risk of detection and degrades the content quality. Therefore, our approach of using stuffing_bytes balances embedding capacity with stealthiness and maintains video integrity without compromising quality or increasing detection vulnerability.

B. Design and Implementation of a Complete Transmission and Reception Framework

The entire transmission and reception process relies on a pre-shared pseudo-random seed, which ensures that both the sender and receiver can correctly locate the TS packet containing the covert information during transmission and extraction, without requiring additional metadata. This enhances the security of the covert channel.

As shown in Figure 3, the sender follows a structured process to embed covert data within TS packets:

1) **Preliminary Setup:** Before transmission, both the sender and receiver agree on a shared pseudo-random seed. This seed is used to deterministically select TS packets for embedding covert information, ensuring synchronization between encoding and decoding.
2) **Covert Message Length Encoding:** The sender first converts the length of the covert message into a 4-byte integer. This length indicator is crucial for the receiver to correctly reconstruct the message.
3) **Pseudo-Random TS Packet Selection:** Using the shared seed, the sender generates a sequence of pseudo-random numbers corresponding to TS packet indices within the HLS stream. The first selected TS packet is used to store the encoded message length.
4) **Message Segmentation via Adaptive Stuffing Allocation:** The available stuffing bytes within each selected TS packet vary. To efficiently distribute the covert message, we employ a randomized segmentation algorithm that takes into account the stuffing byte capacity of each TS packet. The detailed procedure is outlined in **Algorithm 1**.
5) **Stream Transmission:** The modified TS packets, containing both the message length and fragmented covert data, are assembled back into the HLS stream and transmitted to the receiver.

Algorithm 1 Message Segmentation via Adaptive Stuffing

Require: TS_Pkts: List of packet indices with available space
Require: Msg_Len: Total length of the covert message
Require: Seed: Pseudo-random seed for sender and receiver
Ensure: Seg_Map: Mapping of packets to allocated message sizes
Initialize $Remain\_Len \leftarrow Msg\_Len$
Shuffle TS_Pkts using $Seed$
for all (Pkt, Max_Spc) in TS_Pkts (shuffled) **do**
  if $Remain\_Len \leq 0$ **then**
    break
  end if
  $Alloc\_Len \leftarrow Random(1, \min(Max\_Spc, Remain\_Len))$
  Append $(Pkt, Alloc\_Len) \rightarrow Seg\_Map$
  $Remain\_Len \leftarrow Remain\_Len - Alloc\_Len$
end for
if $Remain\_Len > 0$ **then**
  for all unselected (Pkt, Max_Spc) in TS_Pkts **do**
    if $Remain\_Len \leq 0$ **then**
      break
    end if
    $Alloc\_Len \leftarrow \min(Max\_Spc, Remain\_Len)$
    Append $(Pkt, Alloc\_Len)$ to Seg_Map
    $Remain\_Len \leftarrow Remain\_Len - Alloc\_Len$
  end for
end if
if $Remain\_Len > 0$ **then**
  return "Segmentation Failed"
else
  return Seg_Map
end if
=0

The receiver, upon receiving the HLS stream, extracts the hidden message using the following steps:

1) **TS Packet Analysis:** The receiver analyzes incoming TS packets to identify those containing adaptation fields with stuffing bytes available for covert data extraction.
2) **Pseudo-Random TS Packet Selection:** Using the pre-shared seed, the receiver regenerates the same pseudo-random sequence as the sender, identifying the exact TS packets that contain embedded data.
3) **Message Length Extraction:** The first identified TS packet is read to extract the 4-byte length encoding, determining the total number of bytes expected in the covert message.
4) **Fragment Assembly:** The receiver follows the same pseudo-random segmentation pattern to

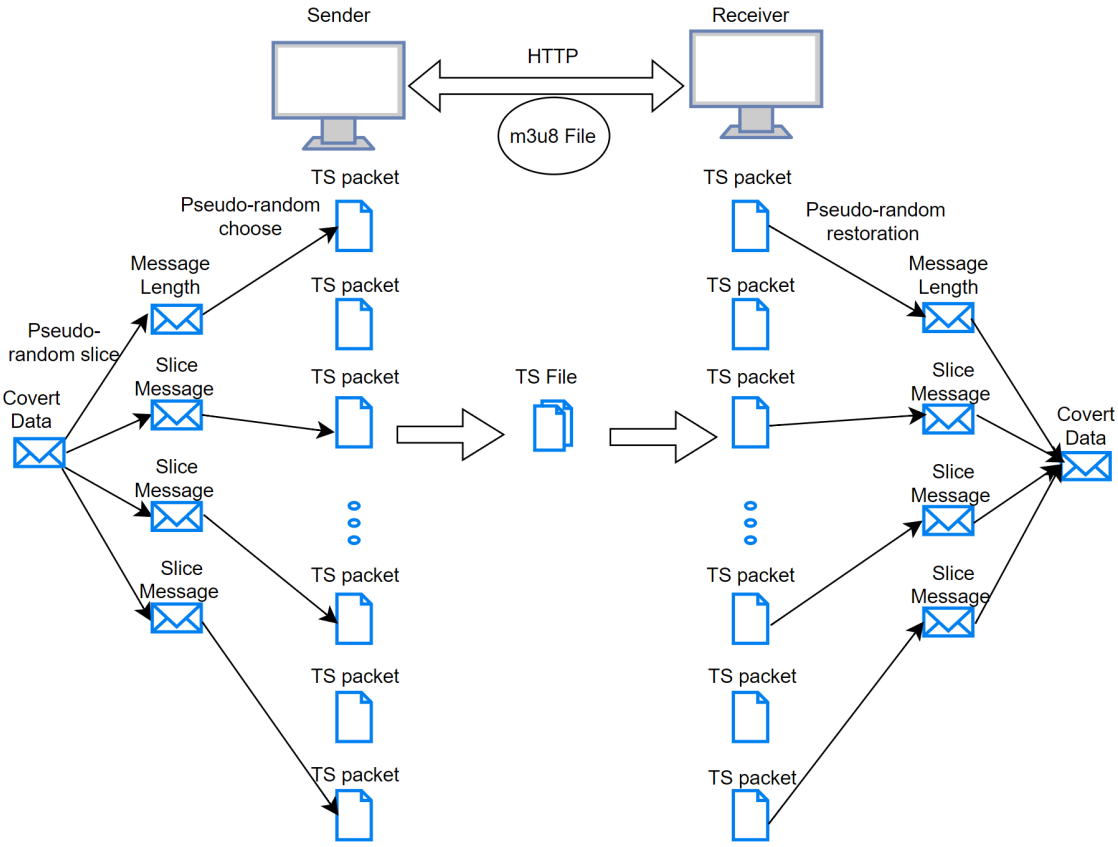

Fig. 3. HLS transmission and reception framework

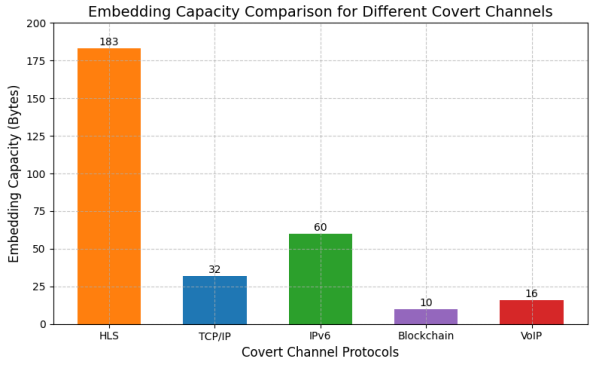

Fig. 4. Comparison of Embedding Capacity

reconstruct the message. By applying the segmentation method described in **Algorithm 1**, the extracted fragments are mapped back to their original sequence, ensuring correct reassembly of the covert message.

5) **Message Decoding and Output:** The final message is reassembled, ensuring data integrity. Any errors due to missing or corrupted packets are handled using redundancy mechanisms if necessary.

C. Dynamic Padding Strategy for TS Packets in HLS

In the previous subsections, we focused on leveraging existing TS packets within the HLS stream for covert data embedding. However, to further expand the covert capacity, we propose a *Dynamic Padding Strategy*, which introduces additional TS packets without disrupting the original playback. This method exploits the fact that HLS operates within a target bitrate, allowing for controlled padding while maintaining the expected bitrate constraints.

To ensure compliance with HLS playback mechanisms:

- Padding packets are designed with appropriate *Program Clock Reference (PCR)* adjustments to maintain proper timing.
- The inserted packets do not disrupt the sequence of audio and video streams.
- The interval for inserting padding packets is dynamically adjusted based on the measured bitrate.

To implement this strategy, we introduce **Algorithm 2**, which computes the necessary padding interval and inserts dynamically generated TS packets accordingly.

In conclusion, instead of embedding a single, continuous block of covert information within the stuffing bytes of a single TS packet, we split the covert message and distribute it across multiple TS packets. This is achieved through the use of a pseudo-randomly generated packet sequence, ensuring that the covert data is scattered across different packets. By doing so, we avoid having large, easily detectable blocks

Algorithm 2 Dynamic Padding Packet Insertion for HLS TS Streams

Require: TS_File, Tgt_Bitrate
Ensure: Output_File with inserted padding packets

Compute $Act\_Bitrate, Duration \leftarrow CalculateBitrate(TS\_File)$
Compute $Total\_Pkts \leftarrow FileSize(TS\_File)/188$
Compute $Interval \leftarrow CalculatePaddingInterval(Act\_Bitrate, Tgt\_Bitrate, Total\_Pkts)$
if $Interval = \infty$ **then**
  No padding needed, terminate
end if
Initialize $Count \leftarrow 0$, $PCR\_PID \leftarrow None$, $PCR\_Val \leftarrow None$
for each Pkt in TS_File **do**
  Write Pkt to Output_File
  $Count \leftarrow Count + 1$
  if Pkt contains PCR **then**
    Update PCR_Val
  end if
  if $(Count \bmod Interval = 0)$ **and** $(PCR\_PID \neq None)$ **and** $(PCR\_Val \neq None)$ **then**
    Create $Pad\_Pkt \leftarrow (PCR\_Val + 1ms)$
    Insert Pad_Pkt into Output_File
  end if
end for
return Output_File =0

of hidden data within a single packet. This random distribution increases the complexity of identifying the embedded information and makes it much harder to detect through simple traffic analysis.

Additionally, we have introduced a dynamic padding mechanism for TS packets. This method allows us to generate more TS packets within a given TS file, effectively increasing the space available for embedding covert data while also introducing variability in the traffic flow. By varying the number and placement of padding packets, we create additional noise within the traffic, further obfuscating the presence of hidden information. This dynamic approach not only improves the capacity of the covert channel but also helps to camouflage the communication, making it more resistant to detection.

It is also important to emphasize that even without the insertion of covert information, the adaptation field and stuffing bytes are inherently present in the TS packets of the HLS protocol. These components are a standard part of the protocol and are used for legitimate purposes, such as synchronization and error correction. Importantly, the adaptation field and stuffing bytes do not conflict with the actual video content payload, as they are separate from the video data. Their presence in the packets is natural and necessary for the proper functioning of the HLS streaming protocol.

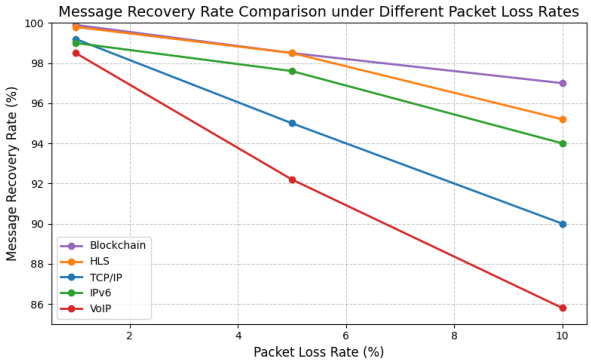

Fig. 5. Comparison of Transmission Robustness under Packet Loss

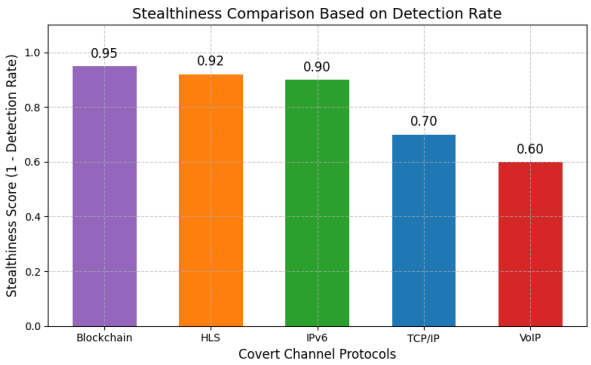

Fig. 6. Comparison of Stealthiness Based on Machine Learning Detection Rate

IV. COMPARATIVE ANALYSIS

In this section, we evaluate the performance of several covert channels, including HLS (HTTP Live Streaming), IPv6, Blockchain, TCP/IP, and VoIP, across various parameters such as embedding capacity, transmission robustness under packet loss, stealthiness, and transmission rate. The results of these evaluations are presented through several figures, each focusing on different performance aspects of these covert channels.

The specific configuration of the experimental environment is as follows:

- **Hardware Environment:** The physical server is equipped with an AMD 5800H CPU and 32GB of memory. VM resources are allocated independently to ensure stability and isolation for each testing platform.
- **Virtualization Platform:** Two independent virtual machines are set up using VMware Workstation Pro, serving as the sender and receiver, with both running Windows 10 as the operating system.
- **Network Configuration:** The VMs are configured with a bridged network mode, using a real router for data forwarding. This setup simulates a real-world public network environment and avoids interference from LAN traffic patterns.
- **Software Environment:** All covert channel implementations are developed in Python 3.11, with

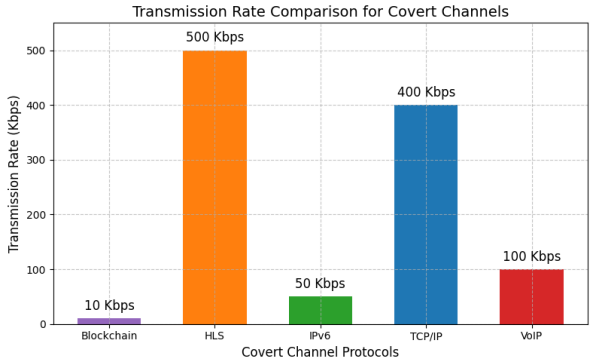

Fig. 7. Comparison of Transmission Rate

consistent library versions across both the sender and receiver platforms.

Furthermore, to simulate the impact of varying network conditions on covert channel performance, the NetEm network emulation tool is employed to control different packet loss rates. This allows for the evaluation of the transmission robustness of each covert channel. All experiments are repeated three times, with the average value taken as the final result to improve the stability and reliability of the data.

As shown in the Figure 4, The embedding capacity measures the maximum amount of covert data that can be embedded into a single packet or data block. This is a critical factor in covert channel, as it determines the potential size of the hidden message. HLS achieves the highest embedding capacity, with 183 bytes of data available for embedding in a single TS packet. The large size of video streams and the use of padding bytes in the TS packets contribute to this high capacity, making HLS an ideal candidate for high-capacity covert channel.IPv6 offers a more modest embedding capacity, ranging from 20 to 40 bytes depending on the size of the extension headers, which are constrained by the path MTU (Maximum Transmission Unit). However, IPv6 provides sufficient capacity for small-scale covert channel, leveraging optional extension headers such as PadN and Destination Options.Blockchain is constrained by the limited size of the transaction data fields. The Bitcoin OP_RETURN field, for example, allows up to 80 bytes of embedded data, though it is commonly limited to 5-10 bytes in practice. This makes Blockchain suitable for embedding small amounts of covert data, but it is far less flexible than HLS. TCP/IP provides a relatively moderate embedding capacity, with the ability to embed 20 to 32 bytes in the header options field or through timing covert channels. This allows for a reasonable level of data embedding while preserving the integrity of the protocol headers. VoIP employs LSB (Least Significant Bit) steganography in G.711 audio frames, which allows for the embedding of 4 to 16 bytes per frame.

As shown in Figure 5, transmission robustness refers to the ability of the covert channel to maintain communication quality under packet loss conditions. Higher values indicate a more resilient channel. Blockchain demonstrates the highest robustness, with a recovery rate of 99.9%. Once a transaction is committed to the blockchain, it cannot be lost, making it highly resilient to packet loss. HLS exhibits good robustness with a recovery rate of 99.8% at low packet loss rates. The buffering mechanism in video players helps recover lost packets, although heavy packet loss may still lead to retransmissions and reduced quality. TCP/IP shows good robustness, with a 99.2% recovery rate at a 1% packet loss rate. The inherent retransmission mechanism in TCP ensures reliable data delivery. IPv6 also has a retransmission mechanism, which makes it more resilient than initially assumed. It shows a 99.0% recovery rate at a 1% packet loss rate, offering comparable robustness to TCP. VoIP has a moderate recovery rate of 98.5% under low packet loss conditions. However, packet loss affects the audio quality, and techniques such as FEC (Forward Error Correction) are employed to mitigate the impact of packet loss on the quality of the transmitted audio.

As shown in Figure 6, to assess the stealthiness of different covert channel techniques, we use a machine learning-based traffic classification approach, where the detection rate serves as an inverse metric for stealthiness. A lower detection rate implies higher stealthiness, meaning the covert traffic is harder to distinguish from normal traffic. Stealthiness is quantified as (1 − Detection Rate), where a higher score indicates greater difficulty in detecting covert channel. In our experiments, Blockchain achieves the highest stealthiness score of 0.95 due to the diversity of its transaction fields, making it challenging for classifiers to detect covert traffic. HLS follows with a score of 0.92, as covert data in TS packets blends seamlessly with regular video traffic, closely resembling normal streaming behavior. IPv6 scores 0.90, thanks to the inconspicuous nature of its extension headers, though some detection tools can still identify anomalies. TCP/IP-based covert channels score 0.70, primarily due to their more detectable header fields, making them easier for classifiers to identify. VoIP scores the lowest at 0.60, as the small changes introduced by LSB steganography in audio frames are detectable by audio feature analysis tools, significantly reducing the stealthiness of the channel.

As shown in Figure 7, the transmission rate measures the speed at which covert data can be transmitted over the channel. A higher rate indicates that more data can be transmitted in a given period, which is critical for real-time applications. HLS achieves a high transmission rate of 500 Kbps, driven by the large bandwidth available for video streams and the relatively large embedding capacity in TS packets. TCP/IP also exhibits a high transmission rate of 400 Kbps, with both header-based and timing-based covert channels enabling efficient data transfer. VoIP achieves a moderate transmission rate of 100 Kbps, which is

constrained by the quality of the audio stream and the low capacity for data embedding within G.711 audio frames. IPv6 has a lower transmission rate of 50 Kbps, as the limited space for embedding covert data within the IPv6 header fields restricts the rate at which information can be hidden.Blockchain has the lowest transmission rate of 10 Kbps, due to the constraints of the blockchain network's transaction frequency and the relatively small size of the transaction fields available for data embedding.

V. CONCLUSIONS

In this paper, we have presented a covert channel framework leveraging the HLS protocol by embedding secret information within the stuffing bytes of TS packets. Our approach not only utilizes existing TS packets for data hiding but also introduces a dynamic padding strategy to expand the covert capacity while ensuring seamless playback. Through a structured transmission and reception framework, we demonstrated the feasibility of embedding and extracting covert messages without violating the constraints of the HLS protocol.

To evaluate our proposed method, we conducted comparative analyses against traditional covert channels, including VoIP-based, TCP/IP-based, IPv6-based, and Blockchain-based covert channel schemes. The experimental results indicate that HLS-based covert channels demonstrate a relatively high resilience against packet loss, benefiting from the buffering mechanisms and redundancy inherent in streaming protocols. Our method achieves a recovery rate of up to 99.8%, showing better performance in maintaining covert data integrity compared to TCP/IP and IPv6, which are more susceptible to packet loss. In terms of embedding capacity, HLS allows for up to 183 bytes per TS packet, which is higher than IPv6 (60 bytes) and Blockchain (5-10 bytes). This results in a higher overall transmission rate, reaching 500 Kbps, while Blockchain and IPv6 exhibit significantly lower rates due to structural constraints. Additionally, the impact on stealthiness against machine learning detection remains minimal, suggesting that HLS-based covert channel is less likely to be detected compared to TCP/IP or VoIP-based methods.

Future work can focus on further enhancing embedding efficiency by integrating advanced encoding techniques and exploring countermeasures against active detection mechanisms. Additionally, we aim to extend our approach to scenarios involving multiple TS files within an HLS video stream, enabling coordinated embedding strategies across segmented media files. This will further expand the covert capacity and improve robustness against detection.

REFERENCES

[1] C. H. Rowland, "Covert Channels in the TCP/IP Protocol Suite," *First Monday*, vol. 2, no. 5, 1997. [Online]. Available: https://doi.org/10.5210/fm.v2i5.528

[2] S. J. Murdoch and S. Lewis, "Embedding Covert Channels into TCP/IP," in *Proc. 7th Int. Workshop on Information Hiding (IH)*, Barcelona, Spain, June 2005, Lecture Notes in Computer Science, vol. 3727, pp. 247–261, Springer.

[3] X. Luo, E. W. W. Chan, and R. K. C. Chang, "TCP covert timing channels: Design and detection," in *Proc. 38th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Anchorage, AK, USA, Jun. 2008, pp. 420–429, IEEE. [Online]. Available: https://doi.org/10.1109/DSN.2008.4630112

[4] F. Iglesias, R. Annessi, and T. Zseby, "DAT detectors: uncovering TCP/IP covert channels by descriptive analytics," *Secur. Commun. Networks*, vol. 9, no. 15, pp. 3011–3029, 2016.

[5] X. Luo, E. W. W. Chan, and R. K. C. Chang, "TCP covert timing channels: Design and detection," in *The 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2008*, June 24-27, 2008, Anchorage, Alaska, USA, pp. 420–429. IEEE Computer Society, 2008.

[6] S. Saha, S. Karapoola, C. Rebeiro, and V. Kamakoti, "YODA: Covert Communication Channel over Public DNS Resolvers," in *53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2023*, Porto, Portugal, June 27-30, 2023, pp. 252–260. IEEE, 2023.

[7] Z. Trabelsi, W. El-Hajj, and S. Hamdy, "Implementation of an ICMP-based covert channel for file and message transfer," in *15th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2008*, St. Julien's, Malta, August 31, 2008 - September 3, 2008, pp. 894–897. IEEE, 2008.

[8] Z. Zhang, X. Zhang, Y. Xue, and Y. Li, "Building a covert timing channel over VoIP via packet length," in *Data Mining and Big Data - 6th International Conference, DMBD 2021, Guangzhou, China, October 20-22, 2021, Proceedings, Part I*, Y. Tan, Y. Shi, A. Y. Zomaya, H. Yan, and J. Cai, Eds. Communications in Computer and Information Science, vol. 1453, Springer, 2021, p

[9] C. Liang, Y.-a. Tan, X. Zhang, X. Wang, J. Zheng, and Q. Zhang, "Building packet length covert channel over mobile VoIP traffics," *Journal of Network and Computer Applications*, vol. 118, pp. 144–153, 2018.

[10] G. Figueira, D. Barradas, and N. Santos, "Stegozoa: Enhancing WebRTC covert channels with video steganography for Internet censorship circumvention," in *ASIA CCS '22: ACM Asia Conference on Computer and Communications Security, Nagasaki, Japan, 30 May 2022 - 3 June 2022*, Y. Suga, K. Sakurai, X. Ding, and K. Sako, Eds., pp. 1154–1167. ACM, 2022.

[11] N. B. Lucena, G. Lewandowski, and S. J. Chapin, "Covert channels in IPv6," in *Privacy Enhancing Technologies, 5th International Workshop, PET 2005, Cavtat, Croatia, May 30-June 1, 2005, Revised Selected Papers*, G. Danezis and D. M. Martin Jr., Eds., vol. 3856, Lecture Notes in Computer Science, Springer, 2005, pp. 147–166.

[12] L. Caviglione, A. Schaffhauser, M. Zuppelli, and W. Mazurczyk, "IPv6CC: IPv6 covert channels for testing networks against stegomalware and data exfiltration," *SoftwareX*, vol. 17, p. 100975, 2022.

[13] J. Wang, L. Zhang, Z. Li, Y. Guo, L. Cheng, and W. Du, "CC-Guard: An IPv6 covert channel detection method based on field matching," in *Proc. 24th IEEE Int. Conf. High Perform. Comput. Commun. (HPCC)*, Hainan, China, Dec. 2022, pp. 1416–1421.

[14] L. Zhang, J. Wang, Y. Guo, H. Zhang, L. Cheng, and W. Xia, "A self-attention mechanism-based model to detect IPv6 multi-field covert channels," *IEEE Trans. Cogn. Commun. Netw.*, vol. 11, no. 1, pp. 258–273, 2025.

[15] Z. Chen, L. Zhu, P. Jiang, C. Zhang, F. Gao, and F. Guo, "Exploring unobservable blockchain-based covert channel for censorship-resistant systems," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 3380–3394, 2024.

[16] T. Zhang, B. Li, Y. Zhu, T. Han, and Q. Wu, "Covert channels in blockchain and blockchain based covert communication: Overview, state-of-the-art, and future directions," *Computer Communications*, vol. 205, pp. 136–146, 2023.