Int. j. inf. tecnol.
https://doi.org/10.1007/s41870-023-01511-4

ORIGINAL RESEARCH

# FIHIM: a framework for information hiding in IPv6 using micro-protocols

**Punam Bedi**[1] · **Arti Dua**[1,2] · **Vinita Jindal**[3]

Received: 14 May 2023 / Accepted: 4 September 2023


**Abstract** Information hiding in network traffic is generally done through network covert channels (NCCs). As Internet Protocol version 6 (IPv6) is replacing Internet Protocol version 4 (IPv4) rapidly, a lot of research is being done on the security aspect of IPv6. Information hiding used with the intention to ensure the privacy of data is a way of assuring information security. Hence, this paper proposes FIHIM, a novel Framework that implements Information Hiding in IPv6 using Micro-protocols. This framework offers two covert channels for hidden communication, IHIM-C, catering to the user's requirements of a higher capacity covert channel, and IHIM-U, a more undetectable covert channel. A prototype of FIHIM was implemented and analyzed over a LAN for three information-hiding characteristics–capacity, robustness, and undetectability. With IHIM-C as the chosen covert channel, a fixed capacity of 32 bits per IPv6 packet was achieved and with IHIM-U as the chosen covert channel, a minimum capacity of 11 bits per IPv6 packet was achieved with enhanced undetectability. Thus, FIHIM can be utilized for IPv6-based covert communications for either higher capacity-based NCC or higher undetectability-based NCC. Furthermore, FIHIM can be used to generate a dataset containing covert communication-based traffic that can be used for developing technologies for detecting IPv6-based NCCs.

**Keywords** Information-hiding · Network covert channel · Network steganography · Internet Protocol version 6 (IPv6) · Protocol steganography · Micro-protocol

## 1 Introduction

Information security is the need of the hour in today's times. In the COVID-19 pandemic where everything was running with the help of the Internet, be it online education, work, professional meetings or personal communications, the dependency on the Internet has increased tremendously [1]. Internet usage has increased many folds during this global pandemic and so has increased the number of cyber-attacks [2]. All organizations want their vital data to be safe from interceptions during network communications. Though cryptography uses encryption to secure the data over the networks, the presence of secret communication is still evident in such transfers. On the other hand, a covert channel/network steganography not only hides the secret data but also hides the existence of secret communication [3]. The covert channel as a means of hidden communication has been used for malicious activities as well as a medium to ensure the privacy of users from firewalls etc. Information hiding using network covert channels to ensure privacy of data, is one way to provide the security of vital data and communication over networks. Network covert channels conceal information in different features of network traffic flows such as redundant fields of protocol header, inter-packet delays etc. A network packet is a basic network data unit. It consists of two major components: network protocol header and payload.

✉ Arti Dua
arti.batra@bcas.du.ac.in

Punam Bedi
pbedi@cs.du.ac.in

Vinita Jindal
vjindal@keshav.du.ac.in

[1] Department of Computer Science, University of Delhi, Delhi, India

[2] Bhaskaracharya College of Applied Science, University of Delhi, Delhi, India

[3] Keshav Mahavidyalaya, University of Delhi, Delhi, India

Published online: 04 October 2023

The network protocol header consists of fields that contain metadata about the packet and the payload part contains the transmitted information. Communication networks use different network protocols to exchange data over networks. The most common and necessary network protocols include Internet Protocol (IP), Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Hypertext Transfer Protocol etc. Out of all the network protocols, the Internet Protocol forms the backbone of today's communication networks. This protocol has two prevalent versions: Internet Protocol version 4 (IPv4) and Internet Protocol version 6 (IPv6). As per Google statistics, about 42.19% of Internet users are accessing Google using IPv6 presently [4]. The transition from IPv4 to IPv6 is slow and is still ongoing [5]. Many covert channels/network steganography techniques have already been proposed for IPv4 in the past [6–10].

Soon IPv6 is expected to take over the Internet. A lot of investigation is already going on the security and privacy aspects of IPv6 [11]. Working on a similar aspect in this paper, various possibilities for designing covert channels using Internet Protocol version 6 were explored and analyzed. For covert communications, it was realized that the requirements of communication may vary from one situation to another. There may be situations in which a journalist might want to upload a set of pictures showing some crime. Whereas, in the other situation, there might be a need of sending a password securely from one end to the other with more focus on undetectability [12]. The former situation focuses on sending high-capacity data covertly whereas the latter situation requires sending less covert data with more focus on undetectability. Hence, in this paper to cater to such situations as per user's need, a novel framework FIHIM (Framework for Information Hiding in IPv6 using Micro-protocols) is proposed that uses two different network covert channels: IHIM-C (Capacity based Information Hiding in IPv6) and IHIM-U (Undetectability based Information Hiding in IPv6) that use IPv6 packets' headers to carry hidden data using *micro-protocols*. IHIM-C creates a covert channel that focuses more on the *capacity* of the covert channel than the undetectability. Whereas IHIM-U focuses more on the *undetectability* of secret data carried through this covert channel. Both of these covert channels make use of their respective internal control protocol, so-called a *micro-protocol.* A *micro-protocol* is an internal control protocol of a covert channel that is used to implement features such as reliability as well as connection management for a covert channel [13]. In FIHIM, the micro-protocol also helps in establishing indirect communication, in the sense that FIHIM utilizes some bits of header fields (chosen for covert communication) of IPv6 for hiding secret data and the remaining bits for storing control information about the secret data. In direct covert communication, all the chosen header fields' bits are used to carry covert data only. Thus, in an indirect communication channel, the adversary needs to know the placement of secret data bits to be able to interpret the message correctly. The use of micro-protocol in IHIM-C helps in session management and indirect communication between the covert sender and the covert receiver. Whereas the use of micro-protocol in IHIM-U additionally helps in increasing the undetectability of covert data as well. To summarize, the key contributions of this paper are as follows:

- FIHIM is a novel framework that caters to two different requirements of a high capacity-based covert channel and a high undetectability-based covert channel that can be selected based on the user's requirements. If a user wishes to use a high-capacity covert channel IHIM-C is chosen, and if he wishes to use a stealthier channel, IHIM-U is used.
- As a part of FIHIM's novelity, both of the proposed techniques, IHIM-C and IHIM-U use micro-protocols with IPv6 as the underlying protocol. To the best of our knowledge, the use of micro-protocols to develop more advanced and realistic covert channels over IPv6 has not been proposed in literature earlier.
- IHIM-C uses micro-protocol for session management and establishing indirect communication between the sender and the receiver. Whereas, in addition to session management and establishing indirect communication, IHIM-U uses micro-protocol for increasing the undetectability of covert data because of the use of randomization in selecting the header fields that carry covert data.
- With the prototype of FIHIM when implemented over a LAN, IHIM-C delivers a fixed capacity of 32 bits per IPv6 packet, and IHIM-U, which supports higher undetectability, assures a minimum capacity of 11 bits per IPv6 packet.

The structure of this paper is as follows. Section 2 describes the Internet Protocol version 6 and micro-protocols. Section 3 gives an overview of related work done in the field of network covert channels/network steganography for covert communications over IPv6. Section 4 elaborates on the proposed FIHIM framework. Further, the algorithms used for the implementation of the prototype of this framework are discussed. In Sect. 5, the development and testing environment, experiments, and the results obtained after conducting the experiments on the prototype of FIHIM are discussed. Further, a comparison of FIHIM with another existing IPv6-based network covert channel tool is done in this section. Section 6 summarizes this work with the conclusion.

## 2 Background

Covert channels have been used as a means of secret communication since the late centuries. To hide information, a network covert channel utilizes/abuses certain characteristics of network traffic flows. Network traffic flows consist of different protocol packets. A network protocol packet is a fundamental unit of the network. Based on how the secret data is stored in a packet, the network covert channels can be categorized into two types: [14]

- Storage-based Network Covert Channels: These covert channels hide data in the storage part (a protocol header, payload, or both) of network packets that are a part of the network flow.
- Timing-based Network Covert Channels: These covert channels hide data in the sequence numbers or delay intervals between subsequent packets which constitute a network flow.

Since the 1980s, many different storage-based network covert channels that use the protocols' headers have been proposed. These covert channels generally utilize the order of packets, random values, and unused or reserved fields of their respective header. Later on, to further enhance the capabilities of network covert channels, covert channel-internal control protocols, so-called *micro-protocols*, were introduced. As the name micro-protocol suggests, these are small-size protocols that exist within the covert storage area of a cover protocol and are used to implement advanced features such as reliability, connection management, etc. for covert channel software. In this paper, two storage-based network covert channels are proposed that use micro-protocols for session management and undetectability. Below, in sub-section 2.1, the header structure of IPv6 protocol is shortly discussed followed by a brief description of micro-protocol in subsection 2.2.

### 2.1 Internet Protocol version 6

The Internet Protocol forms the backbone of the Internet. This protocol is responsible for maintaining logical addresses and routing of packets over the network(s). The most dominant version of this protocol is version 4 which is still widely being used over the internet. IPv4 defines and uses an address length of 32 bits which is too small to cater to the need for unique IP addresses worldwide [15]. To overcome this problem, IPv6 was introduced, which offered 128 bits long IP addresses delivering much larger address space as compared to IPv4. The details and working of version 6 of this protocol are given in Request for Comment (RFC) 8200 [16].

The first field in the IPv6 Header is the version field. It is 4 bits long and contains the version number of the Internet Protocol being used. The value of the version field is 4 for IPv4 and 6 for IPv6. The next field is Traffic Class which is 8 bits long. This field is used for network traffic management. The first six bits of this field are used for Differentiated Services Code Point (DSCP) and the next two bits are used for Explicit Congestion Notification (ECN). The usage of this field is described in RFC 2474 [17] and RFC 3168 [18]. The next field is the Flow Label field. It is 20 bits long and is used by the source to label the sequence of packets belonging to a single flow. The detailed usage of this field is given in RFC 6437 [19]. The next field is the Payload Length field. It is 16 bits long. This contains the length of the payload following the IPv6 header. This length also includes the length of extension headers (if any) attached to the IPv6 header. The next field is the Next Header field which is 8 bits long. This field contains the protocol number of extension headers (if any) attached next to the IPv6 base header. The most commonly known extension headers are hop-by-hop extension header, destination extension header, routing extension header, fragmentation extension header, authentication extension header, and encrypted security payload header. The next field is Hop Limit. It is also an 8 bits long field. It defines the number of nodes a packet can traverse over the network without getting discarded.

The next two fields are Source Address and Destination Address. Both of the fields are 128 bits long and contain 128 bits long logical IPv6 addresses of the sender and receiver of a packet respectively. Any extension header if present follows a base IPv6 header.

### 2.2 Micro-protocol

A lot of studies propose direct network covert channels over different protocols which modify the unused area or reserved field of common network protocol headers. Later a concept of micro-protocols was introduced. A micro-protocol is an internal control protocol of a covert channel that is used to implement advanced features such as reliability, proxy capabilities, simultaneous connections, and connection management of the covert channel. A micro-protocol is a communication protocol, but unlike other communication protocols, the header of a micro-protocol is placed within the hidden data transferred by a covert channel [21]. The word "micro" means small, hence a micro-protocol mostly utilizes a very small and limited area of a covert channel. For understanding and designing a micro-protocol, a prior understanding of the underlying protocol and cover protocol is needed [13]. An underlying protocol in a micro-protocol-supported covert channel is the network protocol utilized for covert communication. For example in the Ping Tunnel tool, the underlying protocol is ICMP (Internet Control Message Protocol) [22].

Next, a cover protocol is the area of the underlying protocol that is used for hidden communication. For the ping tunnel tool, the ICMP echo request payload and ICMP echo reply payload area are used as cover protocol. With this, a micro-protocol can be defined as an internal control protocol that is a part of the cover protocol that controls covert communication. For the Ping Tunnel tool, the micro-protocol used is shown in Fig. 1.

In this paper, the proposed FIHIM makes use of micro-protocols for implementing covert channels using IPv6. The next section presents the related work done by other researchers in the area of network covert channels/network steganography using IPv6.

## 3 Related work

Network covert channels are used to implement information hiding either to ensure the privacy of data from firewalls or to accomplish network attacks in a stealthy manner such as stegomalware. Network Steganography, the youngest classification of steganography also aims at hiding data in network traffic. Many network steganography techniques/network covert channels are proposed in the literature that make use of the most common network protocols like IP [9, 10, 23], TCP [24], ARP (Address Resolution Protocol) [25, 26], NTP (Network Time Protocol) [27, 28].

Further in this section, the techniques that make use of Internet Protocol version 6 for implementing covert channels are discussed.. Atlasis [29] discussed the possibility of abuse of extension headers used in IPv6 to create covert channels. P. Bedi et al. [30] proposed a covert channel using extension headers over a Local Area Network. They suggested the occurrence or nonoccurrence of an extension header in a predecided sequence to transfer a one-bit or a zero-bit value at respective positions. Tian et al. [31] surveyed and classified four categories of covert channels possible in version 6 of Internet Protocol based on reserved values, order, random values, and tunneled traffic. Many other works have done the security analysis of this future-generation protocol. Hendricks et al. [32] suggested that dropping IPv6 packets with Extension Headers makes end-user compromise and also does not add much help in enhancing IPv6 security. Blumbergs et al. [33] proposed two approaches for covert channel development over IPv6 that abuse IPv6 tunnel and dual stack-based transition mechanisms. These mechanisms can bypass Intrusion detection systems and pose a serious threat. Zagar et al. [34] discussed and tested various security issues related to transitions from IPv4 to IPv6 and intrusion detection possibilities in IPv6 using firewalls and Intrusion detection systems. A theoretical analysis of possible covert channels in IPv6 was done by Lucena et al. [35]. They suggested 22 different covert channels theoretically in fields like Traffic Class, Flow Label, Payload Length, Next Header, Hop Limit, Source Address, all well-known extension headers etc. Further, Mazurczyk et al. [20] practically investigated and analyzed the feasibility and actual hiding capacity of network covert channels proposed by Lucena et al. They proposed 6 methods each using a single IPv6 header field from the following fields: Traffic Class, Flow Label, Payload Length, Next Header, Hop Limit, and Source Address fields. The observations on the steganographic capacity of various header fields of IPv6 made by Mazurczyk et al. based on Anonymized Internet Traces 2016 by CAIDA (Center for Applied Internet Data Analysis) [36] are summarized in Table 1.

Zuppeli et al. [37] presented pcapStego, a tool for creating network covert channels within.pcap files. They created covert traffic by using three embedding mechanisms, one using the Traffic class field, the second using the Flow Label field, and the third using the Hop Limit field. In the first embedding mechanism, they used all 8 bits of the Traffic Class field for covert communication which led to too much deviation from the observed average values over the wild internet. In

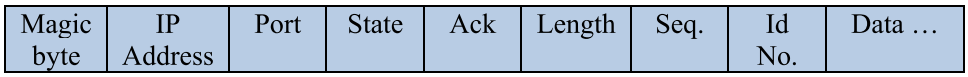


**Fig. 1** Micro-protocol used for ping tunnel [22]

**Table 1** Covert capacity of various IPv6 Header fields as proposed in [20]

| IPv6 header field | Bandwidth for covert channel |
|---|---|
| Traffic class | 2 bits at maximum |
| Flow label | Can be used for steganographic purposes if the values are made pseudo-random using some encryption algorithm |
| Payload length | 10 bits at most (as common MTU size for most Networks is 1500 bytes) |
| Next header | 1 bit (As only TCP/UDP/ICMPv6 extension headers were observed) |
| Hop limit | 1 bit |
| Source address | 128 bits, but highly unreliable as anti-spoofing software can easily detect the manipulations |

the second embedding mechanism, they chose a few active flows from the.pcap file and replaced the Flow label value with covert data. In the third embedding mechanism, 2 different values of Hop Limit fields were used which were 10 for encoding 0 and 250 for encoding 1. The summary of all the above-discussed IPv6-based covert channels is given in Table 2.

These information-hiding techniques are easy to break as one can read the less observed (anomalous) values of the field directly and interpret the messages hidden in these header fields directly. To the best of our knowledge, not much work has been done in developing new advanced and realistic techniques that firstly make use of prospective IPv6 header fields legitimately using the normally observed values for various header fields over the internet and secondly provide a dynamic option to use a more capacity based covert channel or stealthier covert channel for covert communication to the user. So in this work, we propose a novel framework FIHIM, which is a combination of two techniques, IHIM-C and IHIM-U, both of which use micro-protocols to communicate covertly over IPv6 using legitimate header fields' values. This framework works differently in different scenarios: there may be situations in which a journalist wants to send a set of pictures covertly showing some crime. In such a case IHIM-C may be used with more focus on the covert channel's capacity. In the other situation, there might be a need of sending an OTP (one-time password) securely from one end to the other. In this situation, IHIM-U may be used with more focus on undetectability.

A more recent Anonymized Internet Traces 2019 (traffic capture files) dataset provided by the CAIDA [36] was observed and used to select the IPv6 header fields for the design of the covert techniques and micro-protocol. The proposed framework FIHIM and its component covert channels IHIM-C and IHIM-U are described in detail in the next section.

## 4 Proposed FIHIM: a framework for covert communication

FIHIM is a novel framework that implements information hiding in IPv6 protocol using micro-protocols. The proposed framework FIHIM uses two different network covert channels, IHIM-C and IHIM-U, which use IPv6 packets to carry hidden data using micro-protocols. Initially, this section describes the scenario under which the proposed framework was developed and tested. The motivation was taken from the scenario in which two entities named Alice and Bob are captured in a prison and wish to communicate overtly and covertly in the presence of Walter, a warden [38]. If Walter finds out that Alice and Bob are using encryption or are communicating covertly, he would disallow any further

**Table 2** Tabulated literature review of IPv6-based covert channels

| Research work | Technique/IPv6 Field(s) utilized | Advantages | Disadvantages |
|---|---|---|---|
| Atlasis [29] | IPv6 Extension Headers | Can offer high bandwidth | Extension Headers are rarely used over the Internet, hence possible only over a LAN |
| Bedi et al. [30] | IPv6 Extension Headers | High Undetectability | Extension Headers are rarely used over the Internet, hence possible only over a LAN |
| Blumbergs et al. [33] | IPv6 tunnel and dual stack-based transition mechanisms | Robust | Only feasible till IPv6 transition is not complete |
| Lucena et al. [35] | Traffic Class, Flow Label, Payload Length, Next Header, Hop Limit, Source Address, all well-known extension headers etc | Offered a plethora of covert channels that can be developed using IPv6 | All the covert channels are proposed theoretically. Most of the covert channels, for example those utilizing extension headers, source address field are not feasible over the Internet |
| Mazurczyk et al. [20] | Traffic Class, Flow Label, Payload Length, Next Header, Hop Limit, and Source Address | Practically investigated and analyzed the feasibility and actual hiding capacity of the individual header field considered for network covert communication | Proposed the direct use of individual header field as covert channels over the Internet. Use of all 8 bits of Traffic Class for Covert data storage can easily be identified as deviation from normal values |
| Zuppeli et al. [37] | Tool to generate updated packet capture file containing covert data in individual header fields viz. Traffic Class, Flow label, and Hop Limit | Packet capture files with covert data can be obtained on a system without any network dependencies | Need packet capture file as input to generate an updated packet capture file containing covert data. All covert channels are direct and can easily be decoded |

communication between them. So, Alice and Bob need to find a way to communicate using an apparently innocuous channel.

The scenario described above may also have different requirements for covert communication. Like, there may be situations in which a journalist might want to send a set of pictures covertly showing some crime. In the other situation, there might be a need of sending an OTP (one-time password) securely from one end to the other. The former situation focuses on sending high-capacity data covertly whereas the latter situation requires sending less amount of covert data with more focus on undetectability. Thus for these scenarios, two different techniques may be required by Alice and Bob to communicate covertly. Hence, a novel framework FIHIM is proposed that uses two different network covert channels, IHIM-C and IHIM-U as shown in Fig. 2. Both of these covert channels use IPv6 packets with embedded micro-protocol to transfer and manage hidden data.

FIHIM begins with taking as input, the type of communication channel needed from the user as shown in Fig. 2. If a user prefers a high capacity-based covert channel, he inputs 1 and FIHIM uses IHIM-C for covert communication. On the other hand, if the user prefers a higher undetectability-based covert channel, he inputs 2 and FIHIM uses IHIM-U for covert communication. Sections 4.1 and 4.2 gives the detailed working of IHIM-C and IHIM-U respectively.

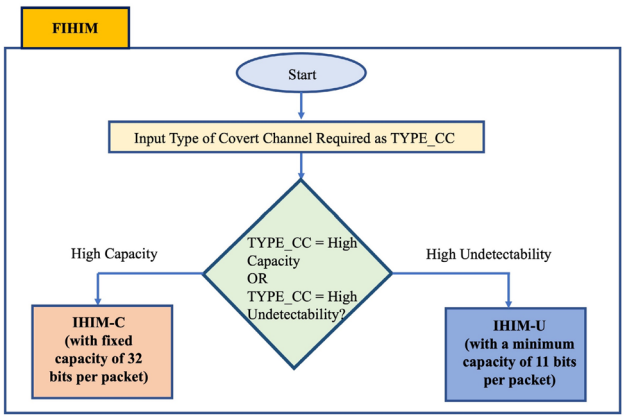


**Fig. 2** Workflow of FIHIM

### 4.1 IHIM-C

IHIM-C is a capacity-based storage network covert channel that uses five IPv6 header fields namely Traffic Class, Flow Label, Payload Length, Next Header, and Hop Limit for covert communication. The Flow Label field containing pseudorandom values can be used for covert communication with a bandwidth of 20 bits [19]. For the Payload Length field, considering the restriction posed by the MTU size value of 1460 (maximum transmission unit) suited mostly for all the networks, the 10 least significant bits out of the 16 bits assigned to the Payload Length field can be used for hidden communication. For Traffic Class, Next Header and Hop Limit fields, the normally observed values of these fields over the wild internet are used for covert communications under IHIM-C. For the Traffic Class field, only three distinct values were observed over the internet 00000000, 00001000, 00001100 [20]. Hence only these three values of the Traffic Class field are used for covert communication. For the Next Header field, the most commonly occurring values are TCP, UDP, and ICMP [20]. Only TCP and UDP values are used for the Next Header field for hidden communication. Similarly, the most commonly observed values for the Hop Limit field are 51–54 and 242–245, hence only two values i.e. 51 and 242 are used for Hop Limit for covert communications under IHIM-C.

IHIM-C uses a total of 34 bits of an IPv6 header for covert communication. Out of these 34 bits, 32 bits are used to carry secret data, and 2 bits are used to carry micro-protocol bits which are used for controlling the covert channel. The segregation of micro-protocol bits and data bits can be done in any combination for the allocated 34 bits of the IPv6 header, but in the current implementation the fifth and sixth bit of the Traffic Class field are used to carry micro-protocol bits and the rest of the fields are used to carry secret data bits as shown in Fig. 3. The micro-protocol bits indicate if the current IPv6 packet with secret data is the last one or if there are more such packets following it. It informs

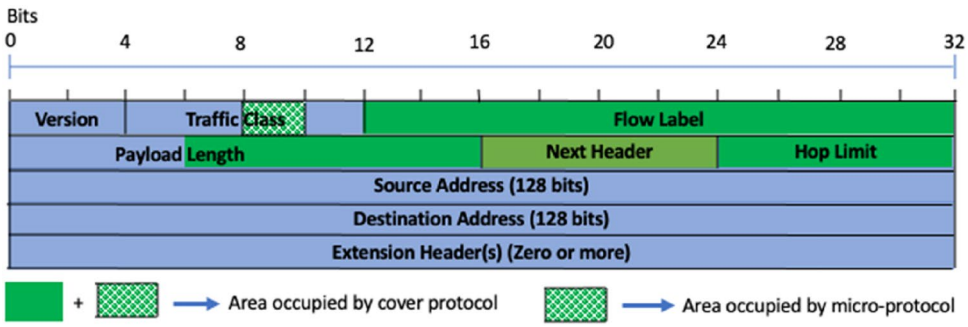


**Fig. 3** Cover protocol and micro-protocol used by IHIM-C in IPv6 header

the receiver when to stop listening for more IPv6 packets. The values and the interpretations of micro-protocol bits are shown in Table 3. The use of micro-protocol adds an advantage of indirect communication between the covert sender and covert receiver as only the sender and receiver know that these two bits carry internal control information and not secret data. Further, the knowledge of interpretations of micro-protocol bits as mentioned in Table 3 is necessary to interpret the secret data correctly which is only known to the covert message receiver and the covert message sender. This further enhances the imperceptibility of secret data.

#### *4.1.1 Sender side*

At the sender's side implementing IHIM-C, initially, a secret message to be sent to the covert receiver is input. The secret message is converted to ASCII, further to hexadecimal, and finally to binary code. The sender then calculates the total number of bits to be sent to the covert receiver. It picks the first 32 bits from the secret message bits to be sent in the first IPv6 synthetic packet. Out of the picked 32 bits, it puts the first 20 bits in the Flow Label field and the next 10 bits in the last ten bits of the Payload Length field. The next one bit is stored in Next Header Field. If this bit is 0, the Next Header field is set to TCP and if this bit is 1, the Next Header field is set to UDP. The last one bit is stored in the Hop limit field. If this bit is 0, the Hop Limit field is set to 51 and if this bit is 1, the Hop Limit field is set to 242.

If the secret message bits are less than 32 bits (which may be the case for the last IPv6 packet), the leftover bits are padded with random bits. The number of the quad(s) (4 bits) used for padding is/are stored in the last quad of 32 bits intended to be used for a secret message. The number of padding bits used includes the bits in the last quad which stores the number of padding bits. The micro-protocol bits (fifth and sixth bit of Traffic Class field), in the case where the IPv6 cover packet carries data with padding, is set to 10. If the secret message bits are greater than or equal to 32 bits then let num_bits_left denote the number of bits left in the secret message after picking the first 32 bits from the secret message for the current IPv6 cover packet.

*Case 1*: If num_bits_left is equal to zero, the micro-protocol bits for the current cover IPv6 packet are set to 11.

**Table 3** Valid values of micro-protocol bits in IHIM-C

| Micro-protocol bits | Interpretation |
|---|---|
| 00 | More IPv6 packets with secret data following the current steganogram |
| 10 | Last IPv6 packet with secret data with padding |
| 11 | Last IPv6 packet with secret data without any padding |

*Case 2*: If num_bits_left is greater than 0, the micro-protocol bits are set to 00.

Dummy payload data is added to the cover IPv6 packet to justify the payload length value. More such IPv6 cover packets are created and sent repeatedly using this technique at a frequency of 1–5 packets per second till the complete secret message is sent. The reason for limiting the frequency of injected IPv6 packets is to cause minimal/no change in network traffic statistics which would further ensure undetectability. These IPv6 cover packets are TCP over IPv6 packets that are sent to the covert receiver with the destination port number set to a fixed value. In our case, we chose a random value of 11,234. The algorithm used at Sender's side for IHIM-C is given in Fig. 4.

#### *4.1.2 Receiver side*

At the receiver's side implementing IHIM-C, the receiver listens for IPv6 packets destined for its IPv6 address with TCP or UDP at the transport layer and destination port set to 11,234. When a packet is received, all 20 bits are fetched from the Flow Label field, and the last 10 bits are fetched from the Payload Length field. The value of the Next Header field is interpreted as 0 if it is a TCP header and 1 if it is a UDP header. Similarly, for the Hop Limit field, a value of 51 is interpreted as a 0, and a value of 242 is interpreted as a 1. These bits are combined and are called combined bits. The fifth and sixth bits of the Traffic Class field carry micro-protocol information. If the received micro-protocol bits are equal to 00, this means more IPv6 packets with secret data are following. The combined 32 bits are added to message bits received in previous IPv6 cover packets (if this is not the first IPv6 packet with secret data). If the received micro-protocol bits value is equal to 10, then as shown in Table 3, this indicates that this is the last IPv6 packet with secret data with some padding in the secret data part (32 bits). In this case, the last quad of combined bits (i.e. bits 29 to 32) contains the number of quads carrying padded data bits including this last quad, this is called num_padding in further references. To fetch the actual data bits from combined bits containing padded data, the number of actual data bits is calculated as follows:

$$\text{temp} = 32 - \left(4 \times (\text{num\_padding})_{10}\right) \tag{1}$$

$$\text{msg\_bits} = \text{combined}\left[0, \text{temp}\right] \tag{2}$$

Next, secret data bits are fetched using Eqs. (1) and (2) and appended to secret data bits received in previous IPv6 packets (if this is not the first IPv6 packet with secret data). If the value of micro-protocol bits is received as 11, then as per Table 3 this indicates it is the last IPv6 packet with secret data in which all the 32 combined bits contain secret

**Input:** A Covert message string S, Destination IPv6 Link-local address Dst_Addr.
**Output:** An IPv6 Packet carrying secret data

1) Encode S: Convert S to ascii, then to hexadecimal and finally to base 2 string, call it SB.
2) msg_length_left = length(SB).
3) While msg_length_left != 0,
   a) If msg_length_left > 32,
      **i)** msg_to_send = SB[0:32]
      **ii)** SB = SB – msg_to_send,
         msg_length_left = msg_length_left – 32
      **iii)** Create an IPv6 packet, pkt.
         pkt.flow_label = msg_to_send[0:20]
         pkt.plen[6:16] = msg_to_send[20:30]
         pkt.traffic_class = 00.
         If msg_to_send[30] == 0, set pkt.next_header =TCP
      Else set pkt.next_header =UDP
         If msg_to_send[31] == 0, set pkt.hop_limit =51
      Else set pkt.hop_limit =241
         pkt.dest_address = Dst_Addr.
         pkt.src_address = Link-local Source IPv6 address.
   b) If msg_length_left == 32, then,
      **i)** msg_to_send = SB[0:32]
      **ii)** SB = SB – msg_to_send.
         msg_length_left = msg_length_left – 32.
      **iii)** Create an IPv6 packet, pkt.
         pkt.flow_label = msg_to_send[0:20].
         pkt.plen[6:16] = msg_to_send[20:30].
         pkt.traffic_class = 11.
         If msg_to_send[30] == 0, set pkt.next_header =TCP
      Else set pkt.next_header =UDP
         If msg_to_send[31] == 0, set pkt.hop_limit =51
      Else set pkt.hop_limit =241
         pkt.dest_address = Dst_Addr.
         pkt.src_address = Link-local Source IPv6 address.
   c) If msg_length_left < 32, then,
      **i)** msg_to_send = SB[0:].
      **ii)** num_pad = (32 – msg_length_left)/4.
      **iii)** Create num_pad number of dummy padding Q**uad**s and store it as pad_data.
      **iv)** combined_bits = concatenate (msg_to_send, pad_data).
      **v)** num_pad_bin = $(num\_pad)_2$
      **vi)** combined_bits[28:32] = num_pad_bin
      **vii)** Create an IPv6 packet
         pkt.flow_label = combined_bits[0:20].
         pkt.plen[6:16] = combined_bits[20:30].
         pkt.traffic_class = 10.
         If msg_to_send[30] == 0, set pkt.next_header =TCP
      Else set pkt.next_header =UDP
         If msg_to_send[31] == 0, set pkt.hop_limit =51
      Else set pkt.hop_limit =241
         pkt.dest_address = Dst_Addr.
         pkt.src_address = Link-local Source IPv6 address.
      **viii)** SB = SB – msg_to_send
      **ix)** msg_length_left = msg_length_left – len(msg_to _send).
   d) Create dummy data (dummy_data) to justify the Payload Length field.
   e) Set Transport Layer Payload = dummy_data.
   f) Set Transport layer destination_port = 11234
   g) send(pkt).
4) Exit.

**Fig. 4** Sender Side algorithm For IHIM-C

data bits without any padding. These bits are fetched and appended to secret message bits received in the previous IPv6 packet with secret data (if this is not the first IPv6 packet with secret data). If the value of micro-protocol bits in the current IPv6 packet is received as 10 or 11, the receiver exits the listening mode and combines all the secret data bits received from the previously processed IPv6 packets, converts the binary message string back to hexadecimal, further to ASCII to read the message string. Figure 5 shows the algorithm used at the receiver's side for implementing IHIM-C.

**Input:** IPv6 Packet with secret data → Pkt
**Output:** Secret message → Secret_msg

1. Initialize MSG_Received = NULL.
2. Listen for IPv6 packets with IPv6.dest_addr = link-local IPv6 address of receiver and TCP.dest_port == 11234.
3. When a packet pkt is received,
   a. Combined_bits[0:20] = pkt.flow_label
   b. Combined_bits[20:30] = pkt.plen.
   c. If pkt.next_header == TCP, set combined_bits[30] =0
      Else set combined_bits[30] =1
   d. If pkt.hop_limit == 51, set combined_bits[31] = 0
      Else set combined_bits[31]=1
4. Cntl_bits = pkt.traffic_class
5. Combined_bits = Combined_bits[0:32]
6. If Cntl_bits == 00000000,
   a. MSG_Received = MSG_Received + Combined_bits.
   b. Goto step 2.
7. If Cntl_bits == 00001000,
   a. num_padding = combined_bits[24:28]
   b. num_padding_dec = $(num\_padding)_{10}$
   c. temp = 32 – (4 * num_padding_dec)
   d. MSG_Received = MSG_Received + combined[0:temp]
   e. Goto step 9.
8. If Cntl_bits == 00001100,
   a. MSG_Received = MSG_Received + combined_bits
   b. Goto step 9.
9. Secret_msg = Decode(MSG_Received)
10. Exit.

**Fig. 5** Receiver Side algorithm for IHIM-C

The communication between the peers here is indirect because of the use of micro-protocol. An adversary who has no knowledge of the usage of micro-protocol in IHIM-C will read all the bits used for covert communication as secret data bits which is not the case as the fifth and sixth bits of the Traffic Class field are used by the micro-protocol. However, if the adversary has knowledge about the placement of secret data bits and the micro-protocol bits in the IPv6 packets then the secret data can easily be decoded. To overcome this limitation, the undetectability aspect needs to be strengthened. Thus, a component of randomness in choosing header fields of IPv6 is added for every IPv6 packet carrying a secret message which gives us the second technique named IHIM-U. The effect of randomization used in IHIM-U is that it refrains this technique from generating a regular pattern which could help in intercepting the secret data. Section 4.2 explains the sender's and receiver's algorithms of IHIM-U.

### 4.2 IHIM-U

IHIM-U is the second covert channel used in the FIHIM framework which offers higher undetectability. It adds a component of randomness in choosing the IPv6 header fields every time an IPv6 packet is created to store the secret message. This strengthens the imperceptibility of secret data. IHIM-U also uses five header fields as cover protocol areas namely Traffic Class, Flow Label, Payload Length, Hop Limit, and Next Header for implementing a covert channel. Out of these, the Flow Label field, the Payload Length

field and the Hop Limit field are used to carry the secret data bits. Whereas the Traffic Class (fifth and sixth bits) and Next Header fields are used for carrying micro-protocol data bits as shown in Fig. 6. To enhance the undetectability of secret data, instead of sending data in pre-defined fixed header fields, a component of randomness in choosing the IPv6 header fields (that will be used for carrying secret data) is added for every IPv6 packet created for carrying secret data. The effect of randomization used in IHIM-U is that it refrains this covert channel from generating a regular pattern which could help in intercepting the secret data. Every time a new IPv6 packet is created a random number is generated which is normalized to a value ranging from 0 to 2 with the help of the modulus function. Each random option value is associated with a pre-defined set of header fields that shall be used to carry covert data. With the use of randomization, this random number information needs to be stored properly in some bits of the cover packet at the sender's side and fetched and interpreted properly from the cover packet at the receiver's side for effective covert communication. To accomplish this IHIM-U uses micro-protocol bits. The micro-protocol area in IHIM-U comprises a fifth and sixth bit of traffic class field and the Next Header field. Only the fifth and sixth bits of the Traffic Class field are used because the most commonly occurring values for the traffic class field in the IPv6 header over the internet are 00000000, 00001000, 00001100 as observed by Mazurczyk et al. [20]. For the Next Header field, the most commonly occurring values are TCP—99.15%, UDP—00.55% and ICMP—0.30% [20]. Only these commonly observed values of the Next Header field are used by the micro-protocol for controlling covert communication. The use of normally occurring values is just to avoid any detection with statistical analysis methods. The micro-protocol used in IHIM-U conveys three significant control information:

1. The location where the secret data is hidden in the IPv6 header.
2. Information about the current IPv6 packet being an intermediate or the last IPv6 packet with secret data.
3. The number of padding bits used in the last IPv6 packet with secret data.

The random option values generated and the corresponding header fields used with or without padding for covert communication are interpreted and used as per the micro-protocol. A tabular form of the same is given in Table 4.

#### *4.2.1 Sender side*

At the sender side implementing IHIM-U, initially, a secret message is input from the user. The message is converted to ASCII, further to hexadecimal, and finally to binary code and stored as message_binary. The sender then calculates the total number of bits to be sent to the covert receiver and stores it as message_length. Until the message_length reduces to zero, the following steps are repeated. Firstly, an IPv6 packet is created. Next, a random value (option) is generated from 0 to 2 to randomly decide the header fields to be used to hide the secret data in the current IPv6 packet. The

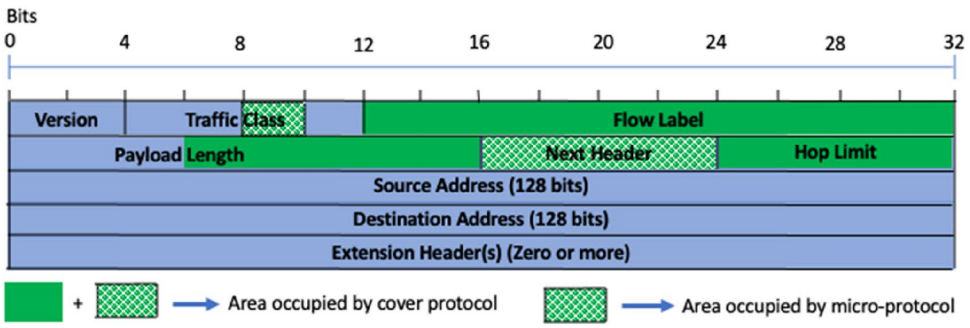


**Fig. 6** Cover Protocol and micro-protocol used by IHIM-U in IPv6 Header

**Table 4** Interpretations of various combinations of valid traffic class field values and Next Header field values for IHIM-U

| Option | Interpretation | DSCP | ECN | Next Header |
|---|---|---|---|---|
| 0 | First/intermediate hidden information in payload length and hop limit fields without padding | 000000 | 00 | TCP |
| 1 | First/intermediate hidden information in flow label and hop limit fields without padding | 000010 | 00 | TCP |
| 2 | First/intermediate hidden information in flow label, payload length and hop limit fields without padding | 000011 | 00 | TCP |
| 0 | Last hidden information in payload length and hop limit fields with/without padding | 000000 | 00 | UDP |
| 1 | Last hidden information in flow label and hop limit fields with/without padding | 000010 | 00 | UDP |
| 2 | Last hidden information in flow-label, payload length and hop limit fields without any padding | 000011 | 00 | UDP |

IPv6 header fields used for hiding secret data are selected based on random numbers generated as per Table 4.

Case I: If the random option generated is 0, then Payload Length and Hop Limit fields are used to store secret data. Next, if the message_length is greater than 11, then the first 11 bits are fetched from message_binary. The first 10 fetched bits are stored in the last 10 bits of the payload length field and the last one bit of fetched 11 bits is stored in the Hop Limit field. If this last bit is 0, the Hop Limit field is set to 51 and if this last bit is 1, the Hop Limit field is set to 242. Further, as per the usage of micro-protocol shown in Table 4, the Traffic Class field is set to 00000000 and the Next Header field is set to TCP (denoting first/intermediate IPv6 packet with secret data). The message_length and message_binary are updated after deducting these 11 bits.

In the other case, if the random option is 0, and message_length is equal to 11, this means that it is the last IPv6 packet with secret data with no padding. Here, the first 11 bits are fetched from message_binary. The first 10 fetched bits are stored in the last 10 bits of the Payload Length field and the last one bit is stored in the Hop Limit field. If this last one bit is 0, the Hop Limit field is set to 51 and if this last one bit is 1, the Hop Limit field is set to 242. And as per the micro-protocol usage shown in Table 4, the Traffic Class field is set to 00000000 and the Next Header field is set to UDP (denoting the last IPv6 packet with secret data). The message_length and message_binary are updated after deducting these 11 bits.

In the other case, if the random option is 0, and message_length is less than 11, this means that it is the last IPv6 packet carrying secret data with some padding bits added. In this case, the dummy random bits are padded before the secret data bits to make it 11 bits long. The number of dummy bits added is stored in the last four bits of the Flow Label field, as this field is currently not being used for secret data transfer in this option. Further, from the combination created from dummy bits (used for padding) and secret data bits, the first 10 bits are stored in the last 10 bits of the Payload Length field and the last one bit is stored in the Hop Limit field. If the last one bit is 0, the Hop Limit field is set to 51 and if the last one bit is 1, the Hop Limit field is set to 242. Further, as per the micro-protocol usage shown in Table 4, the Traffic Class field is set to 00000000 and the Next Header field is set to UDP (denoting the last IPv6 packet with secret data). The message_length is updated to 0 and message_binary is updated to Null.

Case II: If the random option generated is 1, then Flow Label and Hop Limit fields are used to store secret data. If the message_length is greater than 21, then the first 21 bits are fetched from message_binary. The first 20 bits are stored in the Flow Label field and the last bit is stored in the Hop Limit field. If the last one bit is 0, the Hop Limit field is set to 51 and if the last one bit is 1, the Hop Limit field is set to 242. Further, as per the micro-protocol usage shown in Table 4, the Traffic Class field is set to 00001000 and the Next Header field is set to TCP (denoting first/intermediate IPv6 packet with secret data). The message_length and message_binary are updated after deducting these 21 bits.

In the other case, if the random option is 1, and message_length is equal to 21, this means that it is the last IPv6 packet with secret data with no padding. Here, 21 bits are fetched from message_binary. The first 20 fetched bits are stored in the Flow Label field and the last one bit is stored in the Hop Limit field. If the last one bit is 0, the Hop Limit field is set to 51 and if the last one bit is 1, the Hop Limit field is set to 242. Further, as per the micro-protocol usage shown in Table 4, the traffic class field is set to 00001000 and the Next Header field is set to UDP (denoting the last IPv6 packet with secret data). The message_length and message_binary are updated after deducting these 21 bits.

In the next case, if the random option generated is 1 and message_length is less than 21, this means that it is the last IPv6 packet with secret data where padding bits need to be added. In this case, the dummy random bits are padded before the secret data bits to make it 21 bits long. The number of dummy bits added is stored in the last five bits of the Payload Length field, as this field is currently not being used for hiding secret data under this option. Further, from this combination of dummy padding bits and secret data bits, the first 20 bits are stored in the Flow Label field and the last one bit is stored in the Hop Limit field. If the last one bit is 0, the Hop Limit field is set to 51 and if the last one bit is 1, the Hop Limit field is set to 242. As per the micro-protocol usage shown in Table 4, the Traffic Class field is set to 00001000 and the Next Header field is set to UDP (denoting the last IPv6 packet with secret data). The message_length is updated to 0 and message_binary is updated to Null.

Case III: If the random option generated is 2, then Flow Label, Payload Length and Hop Limit fields are used to store secret data. The message_length is greater than 31, then the first 31 bits are fetched from message_binary. The first 20 bits are stored in the Flow Label field, the next 10 bits are stored in the last ten bits of the Payload Length field and the last one bit is stored in the Hop limit field. If the last one bit is 0, the Hop Limit field is set to 51 and if the last one bit is 1, the Hop Limit field is set to 242. As per the micro-protocol usage shown in Table 4, the Traffic Class field is set to 00001100 and the Next Header field is set to TCP. The message_length and message_binary are updated after deducting these 31 bits.

In the other case, if the random option generated is 2, and message_length is equal to 31, it means that it is the last IPv6 packet with secret data without any padding. Here, 31 bits are fetched from message_binary. The first 20 fetched bits are stored in the Flow Label field, the next 10 bits are stored in the last ten bits of the Payload Length field and the

last one bit is stored in the Hop limit field. If the last one bit is 0, the Hop Limit field is set to 51 and if the last one bit is 1, the Hop Limit field is set to 242. Further, the traffic class field is set to 00001100 and the Next Header field is set to UDP (denoting the last IPv6 packet with secret data). The message_length and message_binary are updated after deducting these 31 bits.

In the last case, if the random option generated is 2 and message_length is less than 31, the random number is generated again to choose any one of the previous options. This is done as the last case would require padding in some of the fields (as data bits are less than 31) but there is no field available to store information about the number of padding bits used. Hence this option is not used and a new random number is generated in this special case.

The IPv6 packet cover packets are created and sent repeatedly using this technique at a frequency of 1–5 packets per second till the complete secret information is not sent. The reason for limiting the frequency remains the same, that is, the injected IPv6 packets should cause minimal/no change in network traffic statistics which would further ensure the undetectability of covert communication. The algorithm for the sender side of IHIM-U is shown in Fig. 7.

#### *4.2.2 Receiver side*

At the receiving side using IHIM-U, the covert receiver listens for IPv6 packets destined for its IPv6 address and TCP/UDP destination port set to 11,234. When a packet is received, micro-protocol bits are fetched from the Traffic Class field and Next Header field and interpreted as given in Table 4. If the value of the Traffic Class field is 00000000 and the value of the Next Header is TCP, then as per the micro-protocol used, the secret data is hidden in the last ten bits of the Payload Length field and Hop Limit field.

If the value of the Traffic Class field is 00000000 and the value of the Next Header is UDP, then as per the micro-protocol used, the secret data is hidden in all or some of the last ten bits of the Payload Length field and Hop limit field. The Next Header value of UDP denotes that this is the last IPv6 packet with secret data. The number of paddings used (if any, in the last ten bits of the Payload Length field) is fetched from the last four bits of the Flow Label field. Similarly, if the value of the Traffic Class field is 00001000 and the value of the Next Header is TCP, then as per the micro-protocol used, the secret data is hidden in the Flow Label field and Hop Limit field. If the value of the Traffic Class field is 00001000 and the value of the Next Header is UDP, then as per the micro-protocol used, the secret data is hidden in all or some of the bits of the Flow Label field and Hop Limit field. The number of paddings used (if any, in the Flow Label field) is stored in the last five bits of the Payload Length field. Similarly, if the value of the Traffic Class field is 00001100 and the value of the Next Header is TCP, then as per the micro-protocol shown in Table 4, the secret data is hidden in all the fields using complete twenty bits of the Flow Label field, last ten bits of Payload Length field and Hop Limit.

As per the micro-protocol used, the Traffic Class field value identifies where in the IPv6 header, the secret data is hidden. And the Next Header field value denotes if the current IPv6 packet with secret data is the last one or an intermediate one with more following it. Figure 8 shows the algorithm used to implement IHIM-U at the receiver's side. The next section implements and experiments the working of IHIM-C and IHIM-U over a LAN followed by an analysis of both the covert channels in terms of Capacity, Reliability, and Undetectability.

## 5 Implementation and performance evaluation

FIHIM framework, a combination of IHIM-C and IHIM-U, was implemented as a tool (prototype) and tested over a LAN. The implementation and testing environment consisted of a sender and a receiver that wish to have a hidden communication. Both sender and receiver used their respective link-local IPv6 address for covert communications. Scapy library [39] with Python programming language was utilized to generate and send IPv6 packets with secret data over the LAN. The same was used by the receiver to capture and interpret the covert messages. Wireshark [40], a packet capturing and analysis tool was used to capture incoming and outgoing packets to validate the actual sending and receiving of IPv6 packets with secret data at the sender's and the receiver's end. Sub-section 5.1 discusses the experimental environment and how the experiments were conducted. Sub-section 5.2 discusses the Results received after conducting the experiments for FIHIM over a Local Area Network. This is followed by the analysis of the proposed covert channels in sub-section 5.3. Further, sub-section 5.4 makes a comparison of FIHIM with another IPv6-based network covert channel tool.

### 5.1 Experimental study

The experimental environment for the prototype consisted of Host A, a covert message sender, and Host B, a covert message receiver. They both use their respective link-local IPv6 addresses to communicate with each other over a LAN. FIHIM begins with inputting the user's requirement for a high-capacity channel or a stealthier covert channel. If the user chooses the first option IHIM-C runs. If the user wishes to choose the second option IHIM-U runs. In both cases, the sender first inputs the secret message and converts it into a series of binary bits. These bits are fetched and put into

**Input**: A covert message String S, Destination IPv6 Link-local address Dst_Addr.
**Output**: An IPv6 Packet with secret data

1) Encode S: Convert S to ascii, then to hexadecimal and finally to base 2 string, call it SB.
2) msg_length_left = length(SB).
3) While msg_length_left != 0
   - **a)** Generate a random number Rn between 0 to 2.
   - **b)** If Rn == 0,
     - **i)** If msg_left_length >11,
       - **(1)** msg_to_send = SB[0:11]
       - **(2)** SB = SB – msg_to_send
       - **(3)** msg_length_left = msg_length_left – 11
       - **(4)** Create an IPv6 packet, pkt
         - **(a)** pkt.plen[6:16] = msg_to_send[0:10]
         - **(b)** pkt.plen = pkt.plen + $(20)_2$
         - **(c)** if msg_to_send[10:11] == 0,
           - **(i)** pkt.hop_limit = 51
         - **(d)** else
           - **(i)** pkt.hop_limit = 242
         - **(e)** pkt.traffic_class = 00000000
         - **(f)** pkt.next_header = TCP
         - **(g)** pkt.dest_address = Dst_Addr
         - **(h)** pkt.src_address = Link-local Source IPv6 address
     - **ii)** If msg_left_length == 11
       - **(1)** msg_to_send = SB[0:11]
       - **(2)** SB = SB – msg_to_send
       - **(3)** msg_length_left = msg_length_left – 11.
       - **(4)** Create an IPv6 packet, pkt
         - **(a)** pkt.plen[6:16] = msg_to_send[0:10]
         - **(b)** pkt.plen = pkt.plen + $(8)_2$
         - **(c)** if msg_to_send[10:11] == 0,
           - **(i)** pkt.hop_limit = 51
         - **(d)** else
           - **(i)** pkt.hop_limit = 242
         - **(e)** pkt.traffic_class = 00000000
         - **(f)** R = Pseudo random value 20 bits long.
         - **(g)** pkt.fl = R
         - **(h)** pkt.fl[16:20] = 0000
         - **(i)** pkt.next_header = UDP
         - **(j)** pkt.dest_address = Dst_Addr
         - **(k)** pkt.src_address = Link-local Source IPv6 address
     - **iii)** If msg_left_length < 11
       - **(1)** msg_to_send = SB[0:]
       - **(2)** SB = SB – msg_to_send
       - **(3)** msg_length_left = 0.
       - **(4)** Add random padding bits at the beginning of msg_to_send to make it 11 bits long. Store the number of padding bits added as num_padding.
       - **(5)** Create an IPv6 packet, pkt
         - **(a)** pkt.plen[6:16] = msg_to_send[0:10]
         - **(b)** pkt.plen = pkt.plen + $(8)_2$
         - **(c)** if msg_to_send[10:11] == 0,
           - **(i)** pkt.hop_limit = 51
         - **(d)** else
           - **(i)** pkt.hop_limit = 242
         - **(e)** pkt.traffic_class = 00000000
         - **(f)** R = Pseudo random value 20 bits long.
         - **(g)** pkt.fl = R
         - **(h)** pkt.fl[16:20] = $(num\_padding)_2$
         - **(i)** pkt.next_header = UDP.
         - **(j)** pkt.dest_address = Dst_Addr
         - **(k)** pkt.src_address = Link-local Source IPv6 address
     - **iv)** Create dummy data (dummy_data) to justify the Payload Length field.
     - **v)** Add dummy data created above as payload to Transport Layer Header. Set the destination port to 11234 (any one fixed pre-decided random value).
     - **vi)** send(pkt).
   - **c)** If Rn == 1
     - **i)** if msg_left_length >21
       - **(1)** msg_to_send = SB[0:21]
       - **(2)** SB = SB – msg_to_send
       - **(3)** msg_length_left = msg_length_left – 21
       - **(4)** Create an IPv6 packet, pkt
         - **(a)** pkt.fl = msg_to_send[0:20]
         - **(b)** if msg_to_send[20:21] == 0,
           - **(i)** pkt.hop_limit = 51
         - **(c)** else
           - **(i)** pkt.hop_limit = 242
         - **(d)** pkt.traffic_class = 00001000
         - **(e)** pkt.next_header = TCP.
         - **(f)** pkt.dest_address = Dst_Addr
         - **(g)** pkt.src_address = Link-local Source IPv6 address
     - **ii)** If msg_left_length == 21. Then
       - **(1)** msg_to_send = SB[0:21]
       - **(2)** SB = SB – msg_to_send
       - **(3)** msg_length_left = msg_length_left – 21
       - **(4)** Create an IPv6 packet, pkt
         - **(a)** pkt.fl = msg_to_send[0:20]
         - **(b)** if msg_to_send[20:21] == 0,
           - **(i)** pkt.hop_limit = 51
         - **(c)** else
           - **(i)** pkt.hop_limit = 242
         - **(d)** pkt.traffic_class = 00001000
         - **(e)** pkt.next_header = UDP
         - **(f)** pkt.plen[11:16] = 0000
         - **(g)** pkt.plen = pkt.plen + $(8)_2$
         - **(h)** pkt.dest_address = Dst_Addr
         - **(i)** pkt.src_address = Link-local Source IPv6 address
     - **iii)** If the msg_left_length < 21. Then
       - **(1)** msg_to_send = SB[0:].
       - **(2)** SB = SB – msg_to_send
       - **(3)** msg_length_left = 0
       - **(4)** Add random padding bits at the beginning of msg_to_send to make it 21 bits long. Store the number of padding bits added as num_padding.
       - **(5)** Create an IPv6 packet, pkt
         - **(a)** pkt.fl = msg_to_send[0:20]
         - **(b)** if msg_to_send[20:21] == 0,
           - **(i)** pkt.hop_limit = 51
           - else
           - **(ii)** pkt.hop_limit = 242
         - **(c)** pkt.traffic_class = 00001000
         - **(d)** pkt.next_header = UDP
         - **(e)** pkt.plen[11:16] = $(num\_padding)_2$
         - **(f)** pkt.plen = pkt.plen + $(8)_2$
         - **(g)** pkt.dest_address = Dst_Addr
         - **(h)** pkt.src_address = Link-local Source IPv6 address
       - **(6)** Create dummy data (dummy_data) to justify the Payload Length field.
       - **(7)** Add dummy data created above as payload to Transport Layer Header. Set the destination port to 11234 (any one fixed pre-decided random value).
       - **(8)** send(pkt)
   - **d)** If Rn == 2, then
     - **i)** if msg_left_length >31
       - **(1)** msg_to_send = SB[0:31]
       - **(2)** SB = SB – msg_to_send
       - **(3)** msg_length_left = msg_length_left – 31
       - **(4)** Create an IPv6 packet, pkt
         - **(a)** pkt.fl = msg_to_send[0:20]
         - **(b)** pkt.plen[6:16] = msg_to_send[20:30]
         - **(c)** pkt.plen = pkt.plen + $(20)_2$
         - **(d)** if msg_to_send[30:31] == 0,
           - **(i)** pkt.hop_limit = 51
           - else
           - **(ii)** pkt.hop_limit = 242
         - **(e)** pkt.traffic_class = 00001100
         - **(f)** pkt.next_header = TCP.
         - **(g)**
         - **(h)** pkt.dest_address = Dst_Addr
         - **(i)** pkt.src_address = Link-local Source IPv6 address
       - **(5)** Create dummy data to justify the Payload Length field.
       - **(6)** Add dummy data created above as payload to Transport Layer Header. Set the destination port to 11234 (any one fixed pre-decided random value).
       - **(7)** send(pkt)
     - **ii)** If msg_left_length == 31
       - **(1)** msg_to_send = SB[0:31]
       - **(2)** SB = SB – msg_to_send
       - **(3)** msg_length_left = msg_length_left – 31
       - **(4)** Create an IPv6 packet, pkt
         - **(a)** pkt.fl = msg_to_send[0:20]
         - **(b)** pkt.plen[6:16] = msg_to_send[20:30]
         - **(c)** pkt.plen = pkt.plen + $(8)_2$
         - **(d)** if msg_to_send[30:31] == 0,
           - **(i)** pkt.hop_limit = 51
         - **(e)** else
           - **(i)** pkt.hop_limit = 242
         - **(f)** pkt.traffic_class = 00001100
         - **(g)** pkt.next_header = UDP.
         - **(h)** pkt.dest_address = Dst_Addr
       - **(5)** pkt.src_address = Link-local Source IPv6 address
       - **(6)** Create dummy data (dummy_data) to justify the Payload Length field.
       - **(7)** Add dummy data created above as payload to Transport Layer Header. Set the destination port to 11234 (anyone fixed pre-decided random value)
       - **(8)** Send(pkt)
     - **iii)** If msg_left_length <31
       - **(1)** Go to Step 3).
4) Exit.

**Fig. 7** Sender Side algorithm For IHIM-U

Input: IPv6 packet → pkt
Output: Secret message. → Secret_msg

1) Initialize MSG_Received = NULL.
2) Listen for packets with IPv6.dest_addr == link-local IPv6 address of receiver and TCP.dest_port == 11234.
3) When a packet (pkt) is received,
   a) tc = pkt.traffic_class.
   b) If tc == 00000000,
      i) If pkt.next_header == TCP
         (1) pl = pkt.plen.
         (2) pl_bin = pl – $(20)_2$
         (3) MSG_Received=append(MSG_Received, pkt.plen[6:16].
         (4) if hl == 51
             (a) MSG_Received=append(MSG_Received, 0)
             Else
             (b) MSG_Received=append(MSG_Received, 1).
         (5) Goto step 2).
      ii) Else if pkt.next_header == UDP
         (1) fl = pkt.flow_label
         (2) num_padding = $(fl[16:20])_{10}$
         (3) temp = 10 – num_padding.
         (4) pl = pkt.plen.
         (5) pl_bin = pl – $(8)_2$
         (6) pl_bin = pl_bin[6:16]
         (7) MSG_Received=append( MSG_Received, pl_bin[temp:])
         (8) hl = pkt.hop_limit
         (9) if hl == 51
             (a) MSG_Received=append(MSG_Received, 0)
             Else
             (b) MSG_Received=append(MSG_Received, 1)
         (10) Goto step 4).
   c) If tc == 00001000
      i) If pkt.next_header == TCP
         (1) fl = pkt.flow_label
         (2) MSG_Received=append(MSG_Received, pkt.plen[6:16].
         (3) if hl == 51
             (a) MSG_Received=append(MSG_Received, 0)
             Else
             (b) MSG_Received=append(MSG_Received, 1).
         Goto step 2).
      ii) Else, if pkt.next_header == UDP,
         (1) pl = pkt.plen.
         (2) pl_bin = pl – $(8)_2$
         (3) num_padding = pl_bin[11:16].
         (4) temp = 20 – num_padding.
         (5) MSG_Received=append(MSG_Received, pkt.flow_label[temp:])
         (6) if hl == 51
             (a) MSG_Received=append(MSG_Received, 0)
             Else
             (b) MSG_Received=append(MSG_Received, 1)
         (7) Goto Step 4).
   d) If tc == 00001100
      i) If pkt.next_header == TCP
         (1) fl = pkt.flow_label
         (2) pl = pkt.plen.
         (3) MSG_Received=append(MSG_Received, fl)
         (4) pl_bin = pl – $(20)_2$
         (5) MSG_Received=append(MSG_Received, pkt.plen[6:16]
         (6) if hl == 51
             (a) MSG_Received=append(MSG_Received, 0)
             Else
             (b) MSG_Received=append(MSG_Received, 1)
         (7) Goto step 2).
4) Secret_msg = Decode(MSG_Received)
5) Exit.

**Fig. 8** Receiver's side Algorithm for IHIM-U

IPv6 headers as per the chosen covert channel and its micro-protocol, to create an IPv6 packet with secret data at the sender side. After the IPv6 packets are created, the destination port of the transport header (TCP/UDP) is set to a fixed pre-decided value and the source port is given a random value between 5000 to 50,000. The source and destination address of the IPv6 header of all the packets were set to the link-local addresses of Host A and Host B respectively. For both techniques, the IPv6 packet with secret data was created and sent over the local area network as per the sender side algorithms of the respective covert channel as discussed in Sect. 4. On the receiver's side, the receiver listens for IPv6 packets with the source address set to the link-local IPv6 address of the source, and the destination port set to a fixed pre-decided value (11,234 in our case). This is how the packets with secret data are identified from other IPv6 packets. Once an IPv6 packet with secret data is received, it is processed as per the receiver side algorithm and micro-protocol usage of the respective covert channel to fetch the hidden secret message. The Wireshark tool was installed and used at both ends to confirm the sending and receiving of IPv6 packets for both IHIM-U and IHIM-C.

## 5.2 Results

Results were analyzed through both console output and the Wireshark tool. A secret message, input at Host A was received at Host B successfully using TCP over IPv6 or UDP over IPv6 packets. Figures 9 and 10 show the console window snapshots of the successful transmission of desired IPv6 packets from Host A to Host B using IHIM-C. Figures 11 and 12 show the console window snapshots of the successful transmission of desired IPv6 packets from Host A to Host B using IHIM-U.

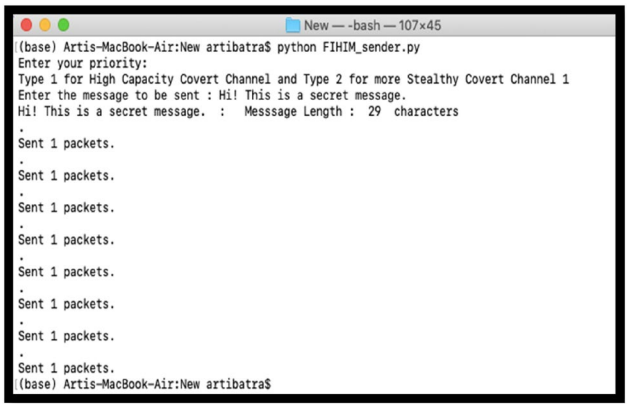


**Fig. 9** Console at Host A using IHIM-C (covert sender)

For IHIM-C, the secret message input at Host A was "Hi! This is a secret message.", as shown in Fig. 9. The length of this message was 29 characters The covert capacity of each packet in IHIM-C is fixed, i.e. 32 bits per IPv6 packet. Hence it required 8 IPv6 packets to send the entered message covertly with this covert channel. A screenshot of Wireshark

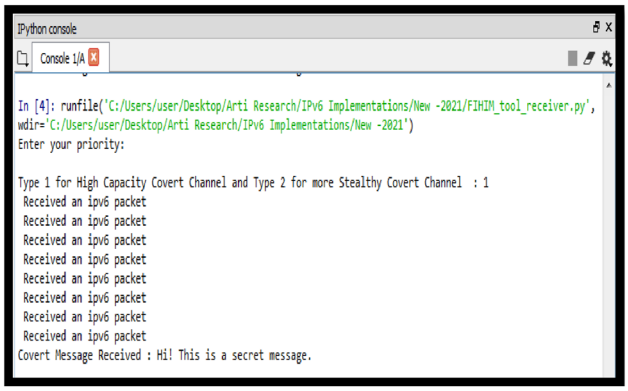


**Fig. 10** Console at Host B using IHIM-C (covert receiver)

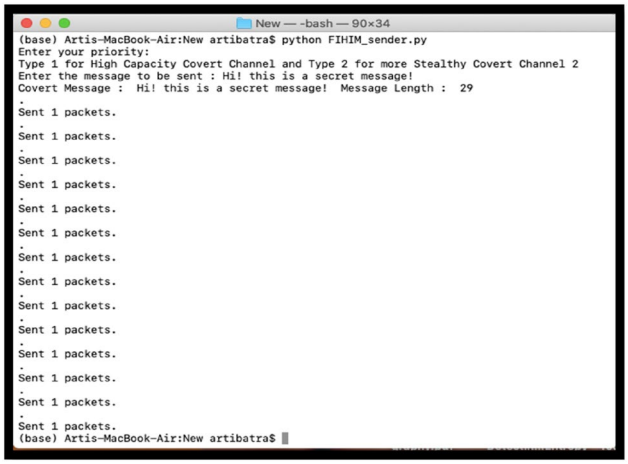


**Fig. 11** Console at Host A using IHIM-U (covert sender)

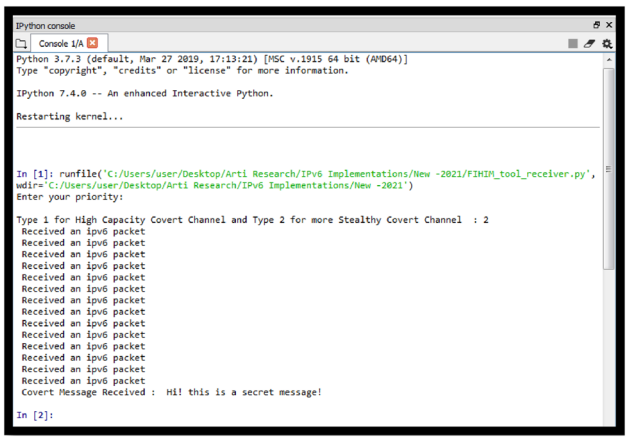


**Fig. 12** Console at Host B using IHIM-U (covert receiver)

capturing TCP over IPv6 packets carrying secret messages from Host A is shown in Fig. 13. Figure 14 shows Wireshark

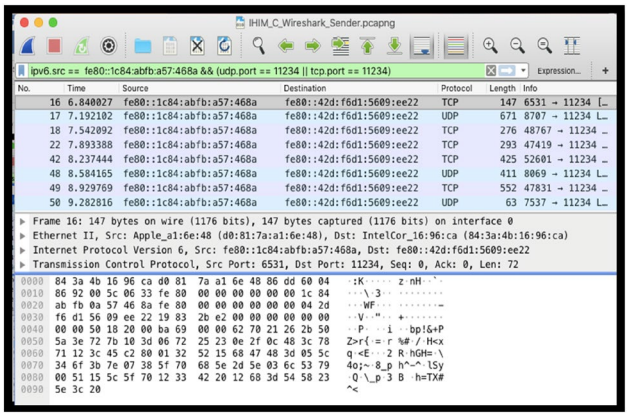


**Fig. 13** Wireshark snapshot at sender site for IHIM-C

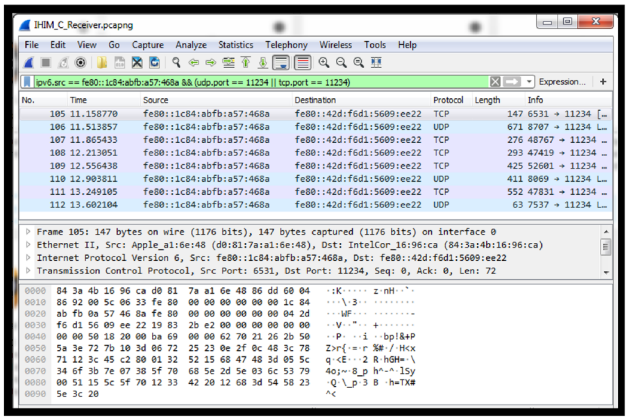


**Fig. 14** Wireshark snapshot at receiver site for IHIM-C

screenshot depicting TCP over IPv6 packets received at Host B.

For IHIM-U, the secret message input at Host A was “Hi! this is a secret message!”. This was also 29 characters long. This covert channel randomly decides which IPv6 header field(s) to use for covert communication. This makes the capacity of this covert channel vary from a minimum of 11 bits per packet to a maximum of 31 bits per packet. In our trial of this covert channel, it took 14 IPv6 packets to transfer this message completely. The average capacity of this trial was 16.57 bits per packet. A Wireshark screenshot of 14 TCP over IPv6 packets and 1 UDP over IPv6 packet used for carrying covert data from Host A to Host B is shown in Fig. 15. Figure 16 shows another Wireshark screenshot that captured 14 TCP over IPv6 packets and one UDP over IPv6 packet received at the receiver side.

Both of the covert channels executed well to give the desired results. In the next sub-section, an analysis is done to understand the efficiency of the proposed covert channels for the three important information-hiding characteristics namely capacity, robustness and undetectability.

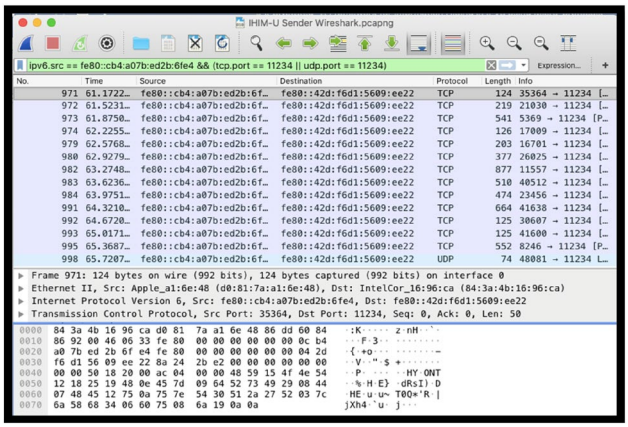


**Fig. 15** Wireshark snapshot at sender site for IHIM-U

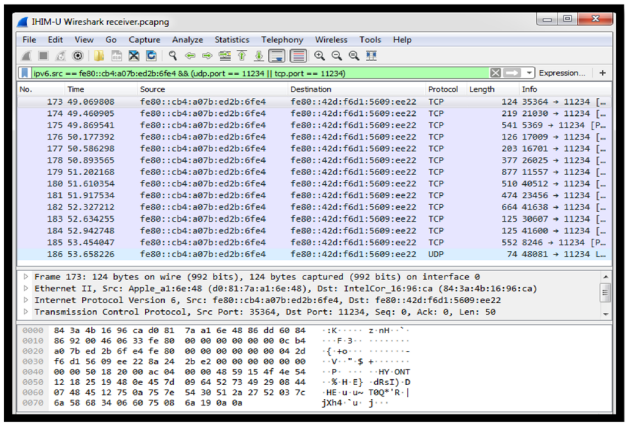


**Fig. 16** Wireshark snapshot at receiver site for IHIM-U

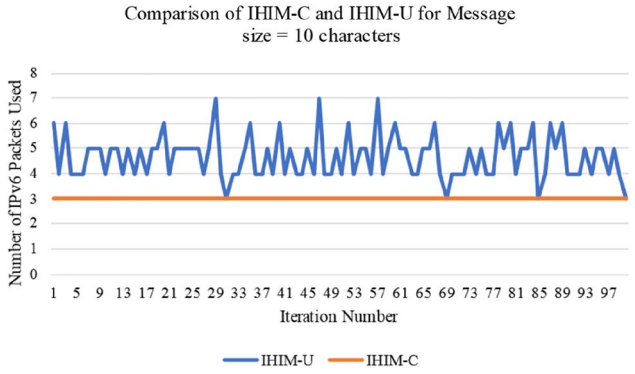


**Fig. 17** Number of Packets used for sending covert messages of message size of 10 characters IHIM-U

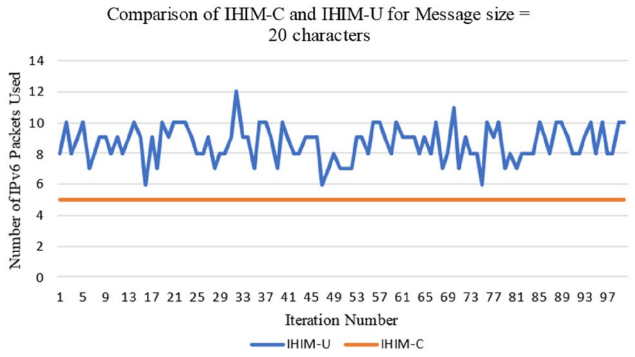


**Fig. 18** Number of Packets used for sending covert messages of message size of 20 characters IHIM-U

### 5.3 Covert channel analysis

Just like other information-hiding mechanisms, the network covert channels are accessed on three characteristics: capacity, robustness, and undetectability. The capacity of a covert channel is defined as the number of covert bits that can be sent in the basic unit of a cover. Robustness defines how resilient a covert channel is to external attacks. Undetectability defines the imperceptibility of secret data in a cover media. In this section, we reviewed the effectiveness of the proposed covert channels IHIM-C and IHIM-U based on these characteristics theoretically or experimentally.

#### *5.3.1 Capacity*

The capacity of a covert channel can be quantized in two ways. It can be defined as the number of secret bits sent per packet or the number of secret bits transferred per unit time. Both the covert channels were compared in terms of the number of packets used by each one of them to send a fixed message. For both the covert channels, the message strings with different message sizes of 10, 20, 30, 40, and 50 characters respectively were sent. For the first covert channel, IHIM-C, there is a fixed capacity of 32 bits per packet, hence it took 3 packets to send a message of size 10 characters. For message sizes of 20, 30, 40, and 50 characters it took 5, 8, 10, and 13 packets to send the respective covert messages. Whereas with the second covert channel IHIM-U, which uses randomness in the selection of header fields, one hundred iterations were performed for sending messages of different sizes of 10, 20, 30, 40, and 50 characters each respectively. Each iteration used a different number of IPv6 packets for sending the complete message. The average number of packets used for messages with message sizes of 10, 20, 30, 40, and 50 characters were found to be 4.68, 8.64, 12.43, 16.44, and 19.69 packets respectively. The detailed observations of 100 trials, for each of the different message sizes with IHIM-U, are shown in Figs. 17, 18, 19, 20, 21.

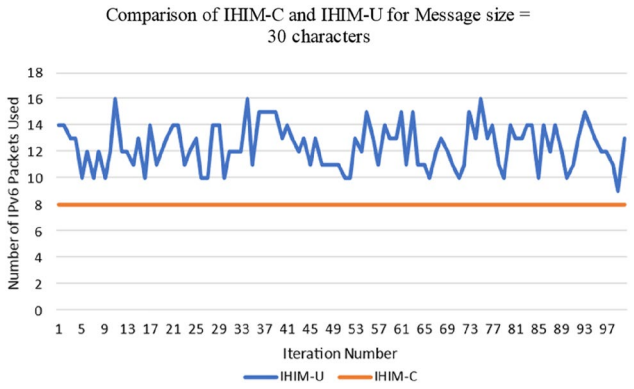


**Fig. 19** Number of Packets used for sending covert messages of message size of 30 characters IHIM-U

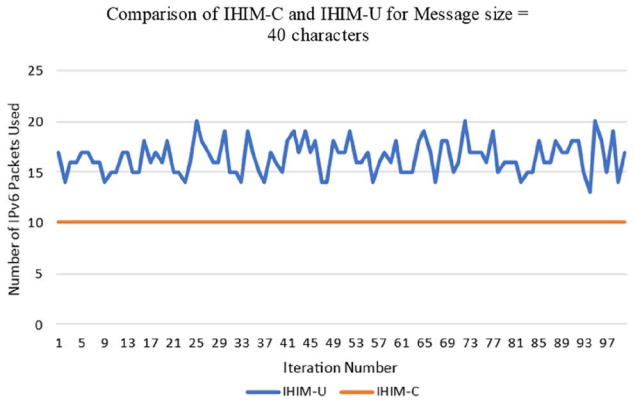


**Fig. 20** Number of Packets used for sending covert messages of message size of 40 characters IHIM-U

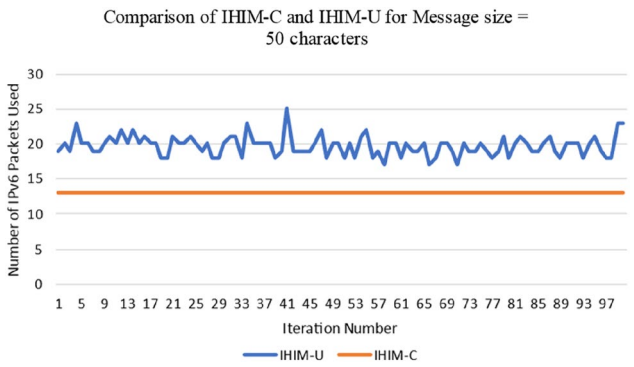


**Fig. 21** Number of Packets used for sending covert messages of message size of 50 characters IHIM-U

### 5.3.2 *Robustness*

Robustness is defined as the resilience of a covert channel against an attack. The robustness of a covert channel can be analyzed as the impact on covert communication if some intermediate node changes some IPv6 header field that is being used for covert transfer. For this, the fields which are vulnerable to changes by intermediate nodes were analyzed. Mazurczyk et al. observed that only the Traffic class field is open to the changes made by intermediate nodes and that the value of DSCP was set to 0b000000 in few cases thereby disrupting the covert channel [20]. No other fields are known to receive changes by intermediate nodes. Thus the use of the Traffic Class field can decrease the robustness in IHIM_U. Whereas, in IHIM_C, only the last IPv6 packet carries a non-zero value for Traffic Class field. Hence, it is majorly robust against normalization attacks.

**Table 5** Commonly observed values of various IPv6 Header Fields as observed by Mazurczyk et al. [20]

| IPv6 header field | Observed Values |
|---|---|
| Traffic class | 00000000, 00001000, 00001100 |
| Flow label | Pseudo-random values |
| Payload length | Variable as per payload size |
| Hop limit | 51–54, 242–245 |
| Next header | TCP–99.15%, UDP–00.55% & ICMP–0.30% |

### 5.3.3 *Undetectability*

A steganographic system is undetectable if it is not possible to distinguish a normal carrier and a carrier with hidden data [41]. To incorporate this in both the proposed covert channels, the values of header fields that are commonly seen over the internet were used. Mazurczyk et al. observed the commonly occurring values for header fields of IPv6 as summarized in Table 5. Also, Anonymized Internet Traces 2019 [36] (traffic capture files) provided by the CAIDA were used to observe the values carried by various IPv6 header fields' over the wild internet.

Further, undetectability was reviewed from two aspects. Firstly, the detection based on single packet inspection is discussed. For this, it was observed how closer the generated IPv6 packets were to the normally observed IPv6 traffic. This was done by calculating the Hamming distance of values used in comparable header fields of the proposed algorithms with values that appear normally in the internet traffic. Hamming distance is a metric that can be applied to two codewords of equal lengths. Hamming distance gives the number of bit positions in which one codeword varies from the other. The Hamming Distance of values used and values observed over the Internet for fields like Traffic Class, Hop Limit and Next Header was calculated. The Hamming distance for two codewords a and b of equal lengths is denoted as D(a, b). For the Traffic Class field, three distances were

computed for each Traffic Class field's value used in the created IPv6 packets as

D1 = D(a,00000000),

D2 = D(a,00001000) and.

D3 = D(a,00001100).

Then we computed hamming distance, HD as below:

HD = D1 & D2 & D3.

If HD is 0, the value is considered normal-appearing value indicating high undetectability. Since only the legitimate values (appearing in the normal internet traffic) were used hence for all the created IPv6 packets, $HD_{\text{Traffic Class}}$ was equal to 0.

With a similar technique, $HD_{\text{Hop Limit}}$ and $HD_{\text{Next Header}}$ were calculated, and for the similar reason that these two fields also used the legitimate values, the corresponding HD value came out to be 0.

Next, the undetectability based on stream data was analyzed. For the normally appearing traffic over the internet, Mazurczyk et al. reported that 99.15% packets indicated the presence of TCP, whereas only 0.55% and 0.3% pointed to UDP and ICMP protocols hence any deviation can easily be spotted as an anomaly [20]. These transport header values were observed in the traffic generated through the proposed covert channels. In IHIM-C, all the packets used TCP at the transport layer. In IHIM-U, only the last packet was sent as UDP over IPv6 packet and for all the previous steganograms TCP was used at the transport layer thereby maintaining the statistical distribution of the Transport layer.

To add further, in a recent work a tool named bccStego was proposed and developed by Repetto et al. [42] that creates usage statistics to reveal covert channels created using three IPv6 Header fields namely, Traffic Class, Flow label, and Hop Limit. They suggested that an IPv6 Covert Channel can be detected with the number of changing bins for all three fields individually. According to them, the number of changing bins would raise abnormally for a respective IPv6 header field if any covert channel is using that particular field. The undetectability aspect of FIHIM as per bccStego is discussed individually for each field below:

*Traffic Class Field*: According to bccStego, the change in the number of bins for the Traffic Class field observed in the network traces provided by CAIDA 2019 dataset can vary from 1 to 7. Any number of changing bins beyond that, which occurs as a result of using all 8 bits of the Traffic Class field to carry covert data can indicate the presence of a covert channel. In FIHIM, both IHIM-C and IHIM-U use only 3 legitimate values of the Traffic Class field, so at any instance, the change in the number of bins can vary from 1 to 3 which makes the use of the Traffic Class field undetectable for covert communication under FIHIM.

*Flow Label Field:* As per bccStego, when an attacker uses the Flow label field to carry covert data the number of changing bins would increase abnormally. As observed by the authors of bccStego for a random trace of 60 min from the CAIDA 2019 dataset, the number of changing bins vary from 0 to 570 within a period of 60 min. This variation happens because of the creation of new flows or the ending of earlier active flows. In FIHIM, the number of new flows added to the existing traffic varies from 1 to 5 per second as the frequency of sending IPv6 packets with secret data under both IHIM-C and IHIM-U can vary from 1 to 5 packets per second. Thus during the interval when covert communication is taking place the change in the number of changing bins can increase maximum by a value of 5 per second which is not a very significant deviation. Thus the use of the Flow label field for covert communication under FIHIM also remains undetectable.

*Hop Limit Field:* For the Hop-Limit field, as observed by Repetto et al. for bccStego, for a random trace of 60 min from the CAIDA 2019 dataset, the number of changing bins can vary from 1 to 19 with legitimate values. In FIHIM, two legitimate values are used for the Hop Limit field to communicate covertly, hence the number of changing bins for the Hop Limit field can vary from 0 to 2, which lies within the observed range.

To summarize, a tabular form depicting the performance of IHIM-C and IHIM-U based on covert channel characteristics is shown in Table 6. Between both of the proposed covert channels, IHIM-C offers a higher and a fixed capacity. This covert channel offers optimum robustness also as there are rare chances of Payload Length or Flow Label fields to be rewritten by the intermediate nodes [19]. The presence of micro-protocol bits in carrier IPv6 packets make the interception harder for the adversary and makes the communication indirect. IHIM-U offers a lesser capacity of a minimum of 11 bits per IPv6 packet but provides higher imperceptibility of secret data because of the randomness in the selection of header fields in IPv6 that carry secret data.

**Table 6** Performance of IHIM-C and IHIM-U based on Information Hiding characteristics

| Technique | Size of Micro-protocol | Capacity | Robustness | Undetectability |
|---|---|---|---|---|
| IHIM-C | 2 bits | 32 bits/packet | High | Lesser than IHIM-U as it only uses micro-protocol bits |
| IHIM-U | 3 bits | 11–31 bits/packet | Lesser than IHIM-C | High, because of randomness |

### 5.4 Comparison with other IPv6-based network covert channel tools

In this section, FIHIM was compared with a recently proposed network covert channel generation tool named pcapStego [37] tested over IPv6. Firstly, FIHIM is developed with an intention of Information Hiding in IPv6 incorporating complete session management of covert communication. Whereas in pcapStego, the intention is to create network covert channels over IPv6 using existing pcap files for emulating attacks or to conduct penetration testing. Secondly, FIHIM creates a fixed number of synthetic IPv6 packets in the network without disturbing the existing network flows. On the other hand, pcapStego overwrites the existing header data for communicating covertly thereby affecting the existing flows in the used pcap file. Thirdly pcapStego uses three direct embedding mechanisms, the first one using all the 8 bits of traffic class field, the second using all 20 bits of Flow Label field and the third using 2 fixed values of Hop Limit field for representing a 0 and a 1 bit. Whereas in FIHIM, two distributed network covert channels are proposed that make use of micro-protocols. One of these can be chosen as per the need of the user for higher capacity (IHIM-C) or higher undetectability (IHIM-U).

(1) IHIM-C uses a combination of two bits of the Traffic Class field, all 20 bits of the Flow Label field, the last ten bits of the Payload Length field, and two legitimate values of each of the Next Header and Hop Limit fields for covert communication. Out of a total of 34 bits, two bits are used for micro-protocol data, and the rest are used for hiding secret data.
(2) IHIM-U implements header field hopping for covert communication. It also uses five IPv6 header fields named Traffic Class, Flow Label, Payload Length, Next Header, and Hop Limit for hiding covert data and micro-protocol bits. It choses randomly from a combination of {Flow Label + Hop Limit} or {Payload Length + Hop Limit} or {Flow Label + Payload Length + Hop Limit} to carry covert data. Traffic Class and Next Header fields are used for carrying micro-protocol bits for covert communication. Lastly, FIHIM uses legitimate values that are observed over the wild internet for fields like Traffic Class, Payload Length, Hop Limit, and Next Header fields which enhances the undetectability of covert communication. On the other hand, pcapStego uses any combination of 8 bits of the Traffic Class field and 2 rarely occurring values of the Hop Limit field (10 and 250) for covert communication which makes the detection of covert communication easy with any AI-based application that is trained on the usual internet traffic data for IPv6.

## 6 Conclusion

With the growth of the Internet, IPv6 is replacing IPv4 rapidly. A lot of work on information hiding in IPv4 has been done by various researchers in the past. Thus in this paper, FIHIM, a novel framework for implementing information hiding in IPv6 using micro-protocols has been proposed. FIHIM offers two covert channels, IHIM-C and IHIM-U which cater to two different requirements of a high capacity-based covert channel and a high undetectability-based covert channel respectively. These covert channels can be selected based on the user's requirements. If a user wishes to use a high-capacity covert channel then IHIM-C is chosen, and if he wishes to use a stealthier channel then IHIM-U is used. Both IHIM-C and IHIM-U create synthetic IPv6 packets with limited frequency without harming existing data flows over the network. FIHIM makes use of legitimate values to execute covert communications to overcome detections based on anomalous values. Both of the proposed techniques, IHIM-C and IHIM-U use micro-protocols with IPv6 as the underlying protocol. IHIM-C uses micro-protocol for session management and establishing indirect communication between the sender and the receiver. Whereas, in addition to session management and establishing indirect communication, IHIM-U uses a micro-protocol for increasing the undetectability of covert data because of the use of randomization in selecting the header fields that carry covert data. Moreover, the use of the Next header field in IHIM-U was done in such a way that it justifies the proportion of TCP and UDP packets, as compared to the statistics observed in the actual wild internet. When experimented over a LAN, IHIM-C provides a fixed capacity of 32 bits per IPv6 packet, and IHIM-U, which supports higher undetectability, assures a minimum bandwidth of 11 bits in a single IPv6 packet. Because of the randomness used in IHIM-U, it offers a lesser average capacity as compared to IHIM-C but surely enhances undetectability. Further, the information-hiding characteristics of the proposed framework were analyzed, in which the capacity, robustness, and undetectability of the proposed covert channels of FIHIM were discussed. It was deduced that IHIM-C offered more capacity whereas IHIM-U offered higher undetectability. Thus, FIHIM can be deployed in real-time as a good candidate for IPv6-based covert communications. Furthermore, this framework can be used to generate datasets containing covert communication-based traffic that can be utilized to develop technologies for detecting IPv6-based covert channels.

**Funding** The authors did not receive support from any organization for the submitted work.

**Data availability** Not applicable.

**Declarations**

**Conflict of interest** The authors have no competing interests to declare that are relevant to the content of this article.

## References


1. Chawla S, Sareen P, Gupta S, Joshi M, Bajaj R (2023) Technology enabled communication during COVID 19: analysis of tweets from top ten Indian IT companies using NVIVO. Int J Inf Technol 15:2063–2075
2. Branscombe M (2021) The network impact of the global COVID 19 pandemic. https://thenewstack.io/the-network-impact-of-the-global-covid-19-pandemic/. Accessed 01 Dec 2022.
3. Pattani K, Gautam S (2021) SonicEvasion: a stealthy ultrasound based invasion using covert communication in smart phones and its security. Int J Inf Technol 13:1589–1599
4. Google (2023) Google IPv6. https://www.google.com/intl/en/ipv6/statistics.html#tab=ipv6-adoption. Accessed 1 May 2023
5. Tomar SS, Rawat A, Vyavahare PD, Tokekar S (2020) Conceptual model for comparison of IPv6 ISPs based on IPv4 traffic profiles. Int J Inf Technol 12:1171–1182
6. Mazurczyk W, Szczypiorski K (2009) Steganography in handling oversized IP packets. Proc Int Conf Multimed Inf Network Secur. https://doi.org/10.1109/MINES.2009.246
7. Kheddar H, Bouzid M (2015) Implementation of steganographic method based on IPv4 identification field over NS-3. Int J Eng Res Appl 5:44–48
8. Mazurczyk W, Smolarczyk M, Szczypiorski K (2011) Retransmission steganography and its detection. Soft Comput 15:505–5151. https://doi.org/10.1007/s00500-009-0530-1
9. Kundur D, Ehsaan K (2003) Practical internet steganography: Data hiding in IP. In: Proceedings of Texas Workshop on Security of Information Systems.
10. Bedi P, Dua A (2020) Network steganography using the overflow field of timestamp option in an IPv4 packet. Proced Comput Sci. https://doi.org/10.1016/j.procs.2020.04.194
11. Chowdhary P, Dwivedi RK (2022) A novel algorithm for traffic control using thread based virtual traffic light. Int J Inf Technol 14:115–124
12. Wendzel S, Keller J (2011) Low-attention forwarding for mobile network covert channels. In: De Decker B, Lapon J, Naessens V, Uhl A (eds) Communications and multimedia security. CMS 2011. Lecture notes in computer science, vol 7025. Springer, Berlin. https://doi.org/10.1007/978-3-642-24712-5_10
13. Wendzel S, Keller J (2012) Systematic Engineering of Control Protocols for Covert Channels. In: De Decker B, Chadwick DW (eds) Communications and multimedia security. CMS 2012. Lecture notes in computer science, vol 7394. Springer, Berlin. https://doi.org/10.1007/978-3-642-32805-3_11
14. Caviglione L (2021) Trends and challenges in network covert channels countermeasures. Appl Sci 11:1641. https://doi.org/10.3390/app11041641
15. Aggarwal M, Kumar N, Yadav K (2017) Survey of named data networks: future of internet. Int J Inf Technol 9:197–207. https://doi.org/10.1007/s41870-017-0014-y
16. Deering S, Hinden R (2017) Internet protocol, version 6 (Specifications). Internet Engineering Task Force. https://tools.ietf.org/html/rfc8200#section-6. Accessed 16 Jan 2023
17. Nichols K, Blakke S, Baker F, Black D (1998) Definition of the differentiated services field (DS Field) in the IPv4 and IPv6 headers. https://tools.ietf.org/html/rfc2474. Accessed 10 Jan 2023
18. Ramakrishanan K, Floyd S, Black D (2001) The addition of explicit congestion notification (ECN) to IP. https://tools.ietf.org/html/rfc3168. Accessed 10 Jan 2023
19. Amante S, Carpenter B, Jiang S, Rajahalme J (2011) IPv6 flow label specification https://tools.ietf.org/html/rfc6437. Accessed 10 Jan 2023
20. Mazurczyk W, Powójski K, Caviglione L (2019) IPv6 Covert channels in the wild. Proc Cent Eur Cybersecur Conf. https://doi.org/10.1145/3360664.3360674
21. Wendzel S, Keller J (2014) Hidden and under control. Ann Telecommun 69:417–430. https://doi.org/10.1007/s12243-014-0423-x
22. Stødle D (2011) Ping tunnel for those times when everything else is blocked. https://www.cs.uit.no/~daniels/PingTunnel/. Accessed 31 March 2023
23. Talbresi Z, Jawahar I (2010) Covert file transfer protocol based on the IP record route option. J Inf Assur Secur 5:64–73
24. Mazurczyk W, Szczypiorski K (2012) Evaluation of steganographic methods for oversized IP packets. Telecommun Syst 49:207–217. https://doi.org/10.1007/s11235-010-9362-7
25. Brodzki AM, Bieniasz J (2019) Yet another network steganography technique based on TCP retransmissions. Proc Int Conf Front Signal Process. https://doi.org/10.1109/ICFSP48124.2019.8938085
26. Bedi P, Dua A (2020) ARPNetSteg: network steganography using address resolution protocol. Int J Electron Telecommun 66:671–677. https://doi.org/10.24425/ijet.2020.134026
27. Schmidbauer T, Wendzel S, Mileva A, Mazurczyk W (2019) Introducing dead drops to network steganography using ARP-caches and SNMP-walks. Proc Int Conf Availab Reliab Secur. https://doi.org/10.1145/3339252.3341488
28. Schmidbauer T, Wendzel S (2020) Covert storage caches using the NTP protocol. Proc Int Conf Availab Reliab Secur. https://doi.org/10.1145/3407023.3409207
29. Atlasis A (1012) Security impacts of abusing IPv6 extension headers. In: Proceedings of Black Hat Security Conference, pp 1–10
30. Bedi P, Dua A (2020) Network steganography using extenson headers in IPv6. In: Badica C, Liatsis P, Kharb L, Chahal D (eds) Information, communication and computing technology. ICICCT 2020. Communications in computer and information science, vol 1170. Springer, Singapore. https://doi.org/10.1007/978-981-15-9671-1_8
31. Tian J, Xiong G, Li Z, Gou G (2020) A survey of key technologies for constructing network covert channel. Secur Commun Netw 2020:1–20
32. Hendriks L, Velan P, Schmidt RT, Boer PT, Pras A (2017) Threats and surprises behind IPv6 extension headers. Netw Traffic Meas Anal Conf (TMA). https://doi.org/10.23919/TMA.2017.8002912
33. Blumbergs B, Pihelgas M, Kont M, Maennel O, Vaarandi R (2016) Creating and detecting IPv6 transition mechanism-based information exfiltration covert channels. In: Brumley B, Röning J (eds) Secure IT systems. NordSec 2016. Lecture notes in computer science, vol 10014. Springer, Cham. https://doi.org/10.1007/978-3-319-47560-8_6
34. Žagar D, Grgić K, Rimac-Drlje S (2007) Security aspects in IPv6 networks–implementation and testing. Comput Electr Eng 33:425–437. https://doi.org/10.1016/j.compeleceng.2007.05.008
35. Lucena NB, Lewandowski G, Chapin SJ (2006) Covert channels in IPv6. Lecture notes in computer science. Springer, Cham, pp 147–166
36. The CAIDA UCSD Anonymized Internet Traces Dataset. [20 Jan 2019, 21 Jan 2019, 22 Jan 2019, 23 Jan 2019]. Center for Applied Internet Data Analysis. https://www.caida.org/data/passive/passive_dataset. Accessed 22 July 2022

37. Zuppelli M, Caviglione L (2021) pcapStego: a tool for generating traffic traces for experimenting with network covert channels. Proc Int Conf Availab Reliab Secur. https://doi.org/10.1145/3465481.3470067
38. Handel TG, Sandford MT (1996) Hiding data in the OSI network model. International workshop on information hiding. Springer, Cham
39. Scapy (2023). Scapy. https://scapy.net. Accessed 1 March 2023.
40. Wireshark (2023). Wireshark. https://www.wireshark.org. Accessed 2 April 2023.
41. Mittelholzer T (1999) An information-theoretic approach to steganography and watermarking. In: Pfitzmann A (ed) Information hiding. IH 1999. Lecture notes in computer science. Springer, Berlin. https://doi.org/10.1007/10719724_1
42. Repetto M, Caviglione L, Zuppelli M (2021) bccstego: a framework for investigating network covert channels. Proc Int Conf Availab Reliab Secur. https://doi.org/10.1145/3465481.3470028