2. BACKGROUND STUDY

One of the most promising and revolutionary extensions of TCP is MPTCP, which uses bandwidth aggregation at the transport layer to get around TCP's drawbacks and provide larger bandwidth and low-latency connections [3]. Moreover, MPTCP offers resilience by employing soft handovers across paths in the event of failure. MPTCP changed the scenario of many industries by enabling the use of multiple disjoint paths over a single connection for communication [28]. With the development of technology in networking, many researchers and industries are participating in research related to issues with MPTCP. Security is one of the crucial issues of MPTCP which attracted focus of many researchers and industries. In this chapter, the overall architecture of MPTCP, working of MPTCP, cryptography and identity-based encryption are discussed in detail which will be useful to understand the proposed research.

2.1 MULTIPATH TCP (MPTCP)

TCP [1], a highly used connection-oriented transport layer protocol, offers a reliable communication between hosts over a packet-switching network. TCP was implemented to offer a process-to-process delivery of packets in the layered hierarchy of TCP/IP protocol suite in which TCP has two interfaces: one to the IP layer and another to the application layer. These interfaces have several functions for data communication and establishing and closing connections between nodes. As shown in Figure 2.1 (b), TCP associates the network interfaces to ensure that only a single IP address can be used during the whole session, leading to connection loss in the event of a communication failure [27]. Suppose the host attempts to transfer the connection through a different network interface while still interacting with the same host. In that case, the ongoing connection will get lost since the host will be connected to a new service provider and bound to a new IP address. Even though, the host get connected with same service provider with the different network interface, the connection will be reestablished from the scratch. Despite the host having several network interfaces, TCP only uses one network interface per connection. Although mobile devices, for instance, feature Wi-Fi and mobile carrier network interfaces, they may use a single interface at a time.

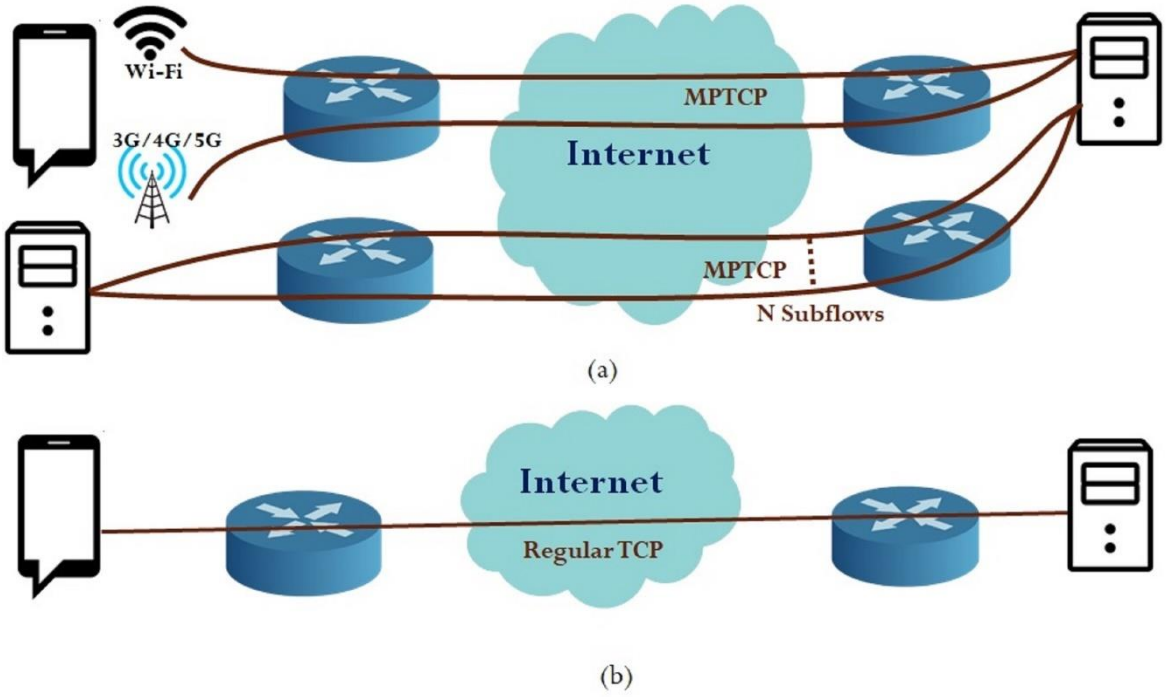

Figure 2.1 (a) MPTCP connection Scenario (b) TCP connection scenario **[27]**

MPTCP was designed by the IETF to overcome inadequacies of TCP and with the goals to enhance throughput, utilization of network resources and resilience of connection against failure through the usage of multiple disjoint paths over single connection [3]. MPTCP is the best alternative for providing traffic aggregation at the TCP level due to its enhanced throughput, TCP friendliness, stable congestion control, and resiliency [29]. MPTCP is built on the top of TCP to offer the backward compatibility and each MPTCP sub-flow corresponds to an individual TCP connection where each sub-flow has own congestion window and roundtrip time [12] [20] [30]. Moreover, in the event of a sub-flow failure or heavy congestion on a given sub-flow, traffic can be redirected to one of the other flows by choosing the least congested TCP sub-flows. MPTCP effectively controls congestion while enabling data transmission via multiple TCP sub-flows. Although there are many options available for congestion control, like the opportunistic linked increases congestion control algorithm (OLIA) [31] and the weighted Vegas (wVegas) [32] for congestion management, MPTCP defaults to the linked increases algorithm (LIA) [33]. Moreover, MPTCP have various packet schedulers like default, round-robin, etc. to divide the data among numerous sub-flows efficiently. MPTCP uses the same data sequence numbering as used by TCP to assurance that packets reach at the receiver end in the correct order [20]. The sub-flow and connection levels are the two layers of data sequences used by MPTCP. Sequence numbers at the connection level depict the total number of packets across all sub-flows. In contrast, sequence numbers

at the sub-flow level are analogous to traditional TCP sequence numbers in that they display the total number of packets on a single sub-flow.

MPTCP connection can be initiated using a 3-way handshake like a traditional TCP connection by exchanging SYN, SYN+ACK and ACK packets. MPTCP uses TCP header with option field as shown in Figure 2.2 so even application-level processes cannot detect whether they are using MPTCP. The application uses a standard TCP socket, and the kernel look after the multipath capability to initiate an MPTCP connection. Initially, the MPTCP establishes a standard TCP connection between the hosts and will use any additional interfaces to connect. The MPTCP-specific information is stored in the TCP header option "Kind" to work with TCP-aware middleboxes and applications [21]. All MPTCP activities are notified via TCP header fields that are optional. Internet Assigned Number Authority (IANA) has given a single TCP option number, "Kind," to MPTCP, and different information can be designated using "subtype" whose values are likewise maintained by IANA repository [3]. Table 2.1 demonstrates the various TCP header subtype used by MPTCP to perform the various sub-flow-related activities.

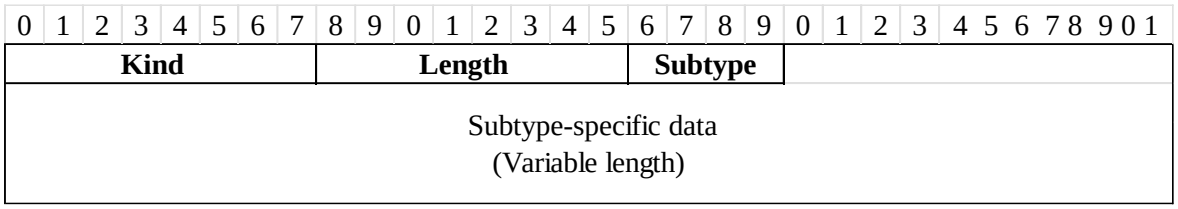

Figure 2.2 MPTCP header format

When an MPTCP option is referred to by its symbolic name, such as "MP_CAPABLE," it denotes to a TCP option with the subtype value of the same as shown in Table 2.1. This subtype is a 4-bit field containing the initial 4 bits of the option payload, as depicted in Figure 2.2.

Consider a host-Alice attempting to connect to a server-Bob, with an eth0 network interface. Host Alice has two network interfaces named eth0 and eth1. Each of these interfaces must have its address (multi-homed capability). As shown in Figure 2.3, initially host Alice uses eth0 to initialize a connection with host Bob over interface eth0. Host Alice will attempt to connect to Bob again via the eth1 interface after the connection has been established successfully. This attempt is referred to as a sub-flow. Figure 2.3 shows the eternal process of MPTCP [22].

Table 2.1 MPTCP Option Subtype in TCP header

Value	MPTCP Option Symbol	Name	Purpose
0x0	**MP_CAPABLE**	Multipath Capable	This option is enabled in SYN, SYN+ACK, and ACK packets at the time connection establishment to indicate the support of MPTCP by communicating host.
0x1	**MP_JOIN**	Multipath Join	It establishes and joins a new sub-flow with the ongoing communication.
0x2	**DSS**	Data Sequence Signal	Used for Data acknowledgment and Data sequence Mapping
0x3	**ADD_ADDR**	Add Address	It is used to advertise newly available network addresses during the connection lifetime.
0x4	**REMOVE_ADDR**	Remove Address	It is used to remove the previously added network address.
0x5	**MP_PRIO**	Change sub-flow priority	It is used for changing the priority of a sub-flow to redirect the data to a specific sub-flow.
0x6	**MP_FAIL**	Fallback	It is used to reset specific sub-flow and immediately fall back to the TCP connection when any middlebox lacks MPTCP support.
0x7	**MP_FASTCLOSE**	Fast close	A connection is quickly ended by sending the MP_FASTCLOSE signal to the peer, and no further data is accepted.
0x8	**MP_TCPRST**	Sub-flow reset	The MP_TCPRST option indicates the reason for delivering an RST on a sub-flow, which can assist implementation in deciding whether to attempt reconnection later.
0xf	**MP_EXPERIMENTAL**	Reserved for private use	Reserved for private use

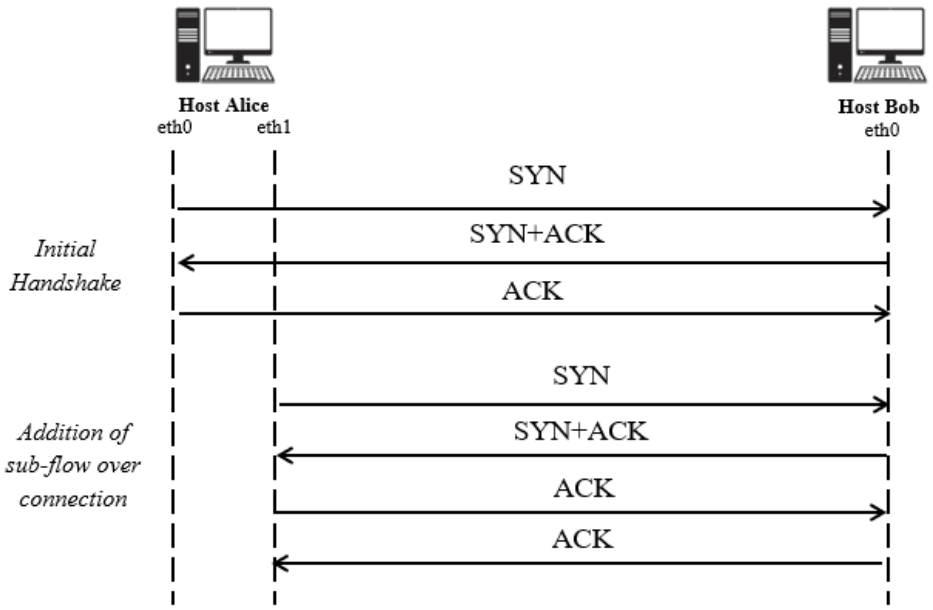

Figure 2.3 MPTCP connection process

2.1.1 MPTCP CONNECTION INITIATION

To initiate the MPTCP connection, MPTCP employs the MP_CAPABLE option with the normal TCP SYN, SYN/ACK, and ACK packets. When an SYN packet with the MP_CAPABLE option is transmitted from sender to receiver, it signals that the sender is MPTCP-compliant. The SYN/ACK packet would include the MP_CAPABLE option if the receiving end is MPTCP-compliant. The sender will then transmit the MP_CAPABLE packet alongside the ACK packet to confirm the MPTCP connection. In addition, to indicate compatibility, MP_CAPABLE will also carry out other crucial tasks. After successfully establishing the MPTCP connection, it must create extra sub-flows connected to the initial connection [3] [22]. To authenticate the extra sub-flows to join them on the initial connection, MPTCP communicates the keys and flags during the initial handshake [22]. MPTCP produces a 64-bit key value for every host, which is to be communicated during the initial handshake. As shown in Figure 2.4, the initial 4-bits of the first octet of the MP_CAPABLE option in TCP header describe the MPTCP option subtype (Here, for MP_CAPABLE, the value is 0, as shown in Table 2.1), while the succeeding four bits indicate the MPTCP version being used (Currently, MPTCP version 0 is used).

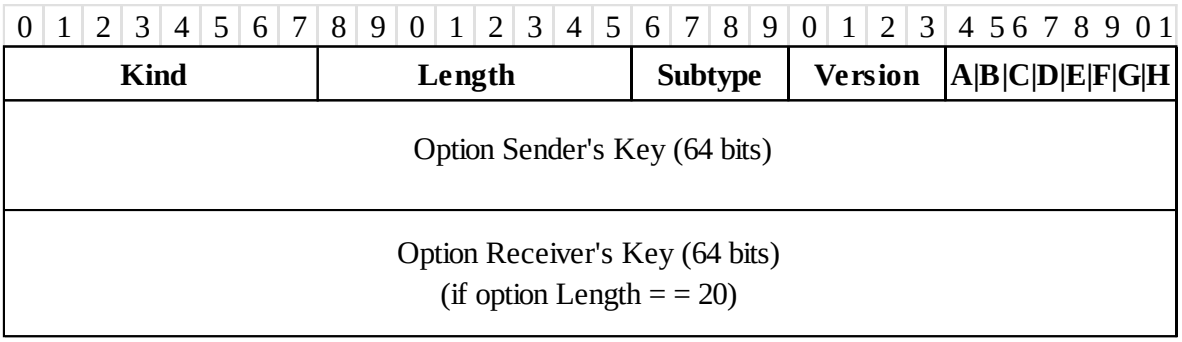

Figure 2.4 Multipath Capable (MP_CAPABLE) Option

The value of "A" bit on the very left must be 1 to signify "Checksum Needed," except the system administrator has precisely agreed on checksums avoidance (maybe because the environment is tightly regulated and there are no middleboxes that could alter the payload).

The "B" bit, which is the second, is an extensibility flag that needs to have value 0 in all present applications. It will be utilized in a forthcoming specification for extensibility purpose. If the value of 'B' flag is 1 in the received SYN packet, it should be discarded immediately by the sender and sender needs to retransmit the packet with standard specification means b flag should be set to 0. Hence, if you set B=1, the MP_CAPABLE option will be longer, and bits C through H will have different meanings.

The left-over bits, D through H, are kept in discussions of cryptographic algorithms. As of now, only the last bit (H) on the right side is used for now. The H-bit denotes that HMAC-SHA1 is used for the encryption. The value of Bit H must be kept 1, and bits C through G must be 0 in an implementation that only supports this method.

Alice and Bob are assumed to be two communicating hosts. Alice possesses two network interfaces, eth0, and eth1. Alice sends the SYN packet by enabling MP_CAPABLE option to the host Bob's eth0 interface, as shown in Figure 2.5. The key K_{Alice} is equipped in this packet. If host Bob is MPTCP compliant, it will respond to SYN packet with MP_CAPABLE from host Alice with an SYN/ACK packet containing the MP_CAPABLE option with host Bob's key K_{Bob}. After Bob confirms MPTCP compatibility, the host Alice, will transmit the ACK packet with the keys K_{Alice} and K_{Bob} to confirm the connection [3]. These keys K_{Alice} and K_{Bob} will be used in future for establishment of subsequent sub-flows between the connection between Alice and Bob. Figure 2.6 and 2.7 shows the Wireshark capture of the MPTCP packet set with MP_CAPABLE option.

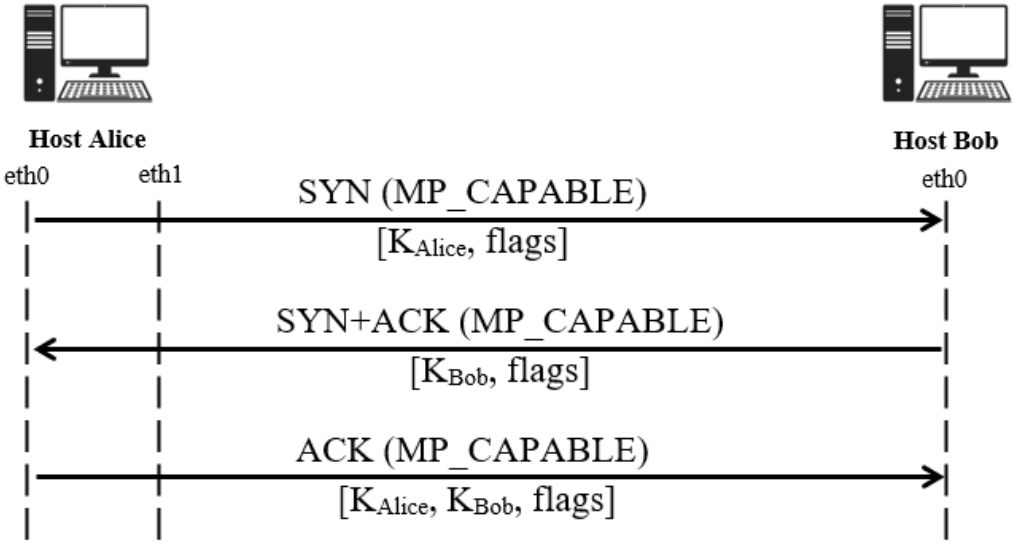

Figure 2.5 MPTCP 3-way handshake process for connection establishment

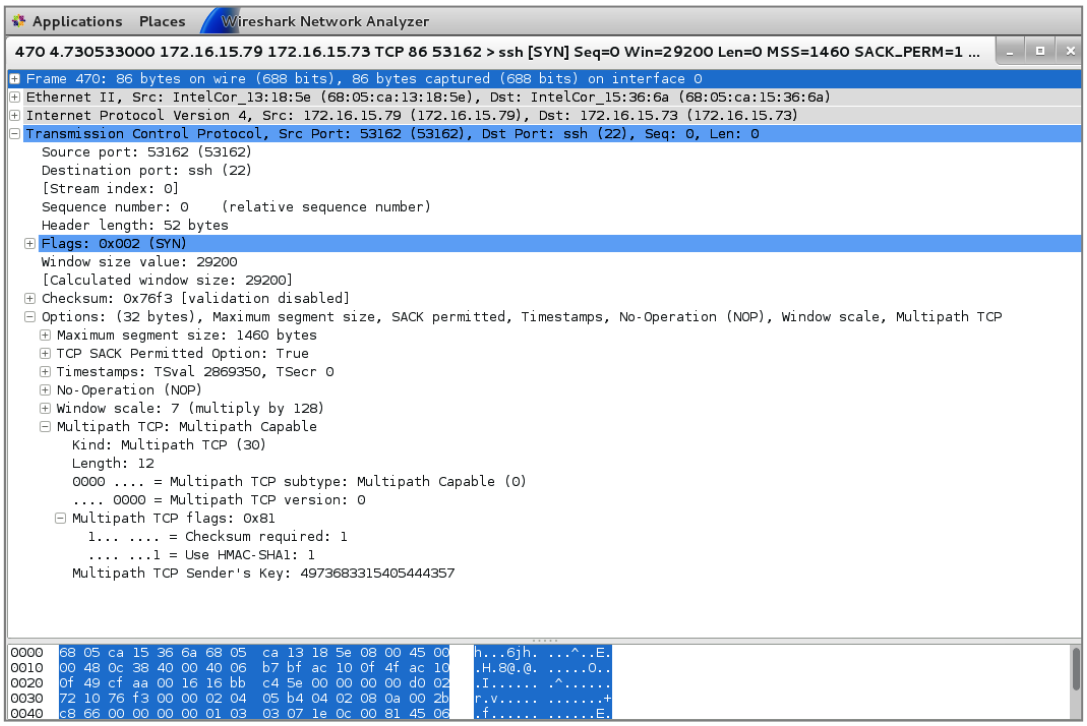

Figure 2.6 Wireshark Capture of MPTCP SYN Packet

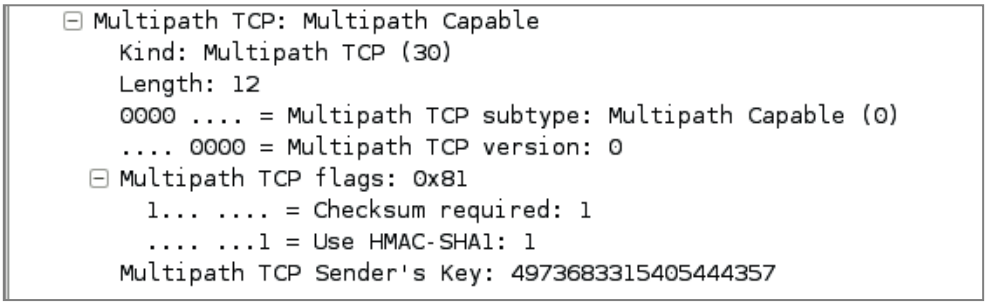

Figure 2.7 MP_CAPABLE option in TCP SYN

2.1.2 JOINING A NEW SUB-FLOW TO THE ONGOING CONNECTION

The main motive behind using MP_CAPABLE option is to ensure that communication hosts are MPTCP-aware and compatible. After establishment of connection between hosts, another sub-flow is established by enabling MP_JOIN option and shared keys [3] [22]. The traditional 3-way handshake method of MPTCP is utilized for starting a new sub-flow, but the MP_JOIN option must be enabled instead of the MP_CAPABLE.

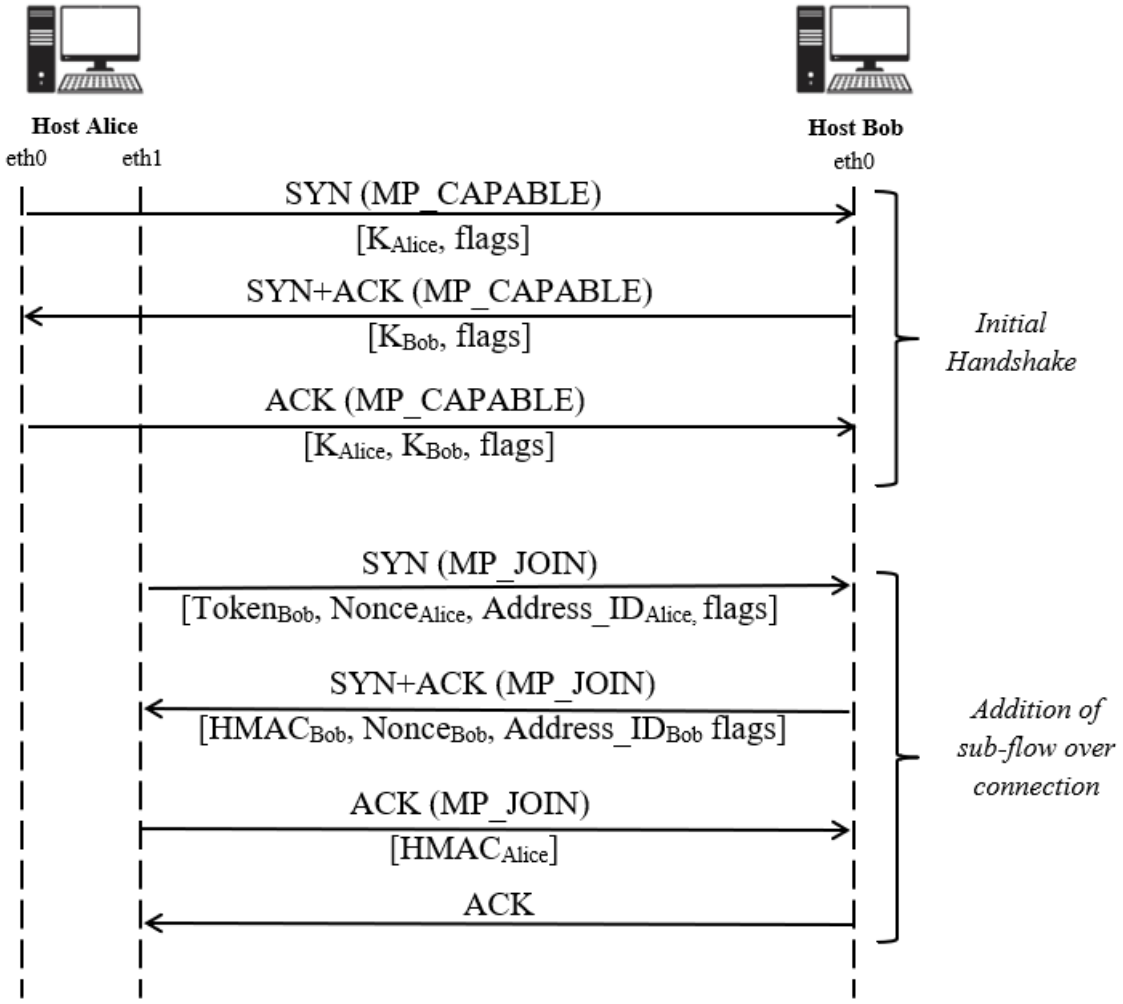

Figure 2.8 MP_JOIN Option to initiate a new sub-flow over the existing connection

MP_JOIN option uses 4 packets to establish a new sub-flow with the ongoing connection, as shown in Figure 2.8. The sender will include HMAC, address ID, and nonce, in SYN packet by enabling MP_JOIN as shown in the packet header in Figure 2.9. The token is a hash produced from the keys shared during connection establishment process using SHA-1, and the first most significant 32-bits will be considered. To fit under the 32-bit token size restriction of the TCP header, we must lower the token's original size. The random number is

included to restrict the replay attack on the authentication method. An “Address ID” is included in the MP_JOIN configuration option so even if a middlebox has altered the IP header, this identifier will still be used to determine the originating address of this packet within the context of the current connection. The Address ID enables address removal over Network Address Translation (NAT)s without requiring knowledge of the source address at the recipient. In case, an MP_JOIN and ADD_ADDR are transmitted at the same time, the Address ID will correlate two events, preventing the setup of two identical sub-flows on the same path [3]. Assume that host Alice has established the connection with host Bob and shared keys K_{Alice} and K_{Bob} with each other as discussed in previous example. Host Alice must send the hash value generated from Bob’s key K_{Bob} along with MP_JOIN option to start a new sub-flow. Bob will authenticate a received packet from host Alice by verifying the token. Here, the address ID recognizes the sender if the middleboxes change the original address.

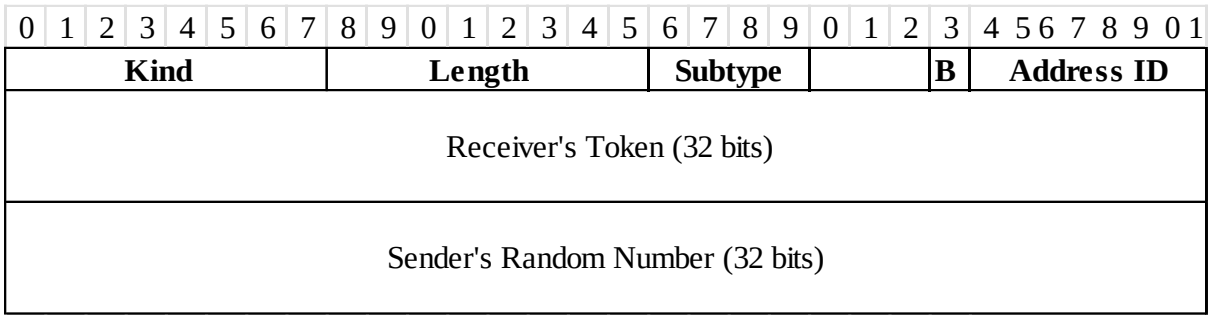

Figure 2.9 Join Connection (MP_JOIN) Option (for Initial SYN**) [3]**

When the first sub-flow in a connection is initiated with an SYN exchange, the Address ID of the sub-flow implicitly specified with value 0. A host must keep its own and the remote host's Address ID/address mappings in its local memory for all the established sub-flows. If an address is deleted from a local or remote host, an implementation must keep track of the previously established sub-flows and the corresponding Address IDs [3].

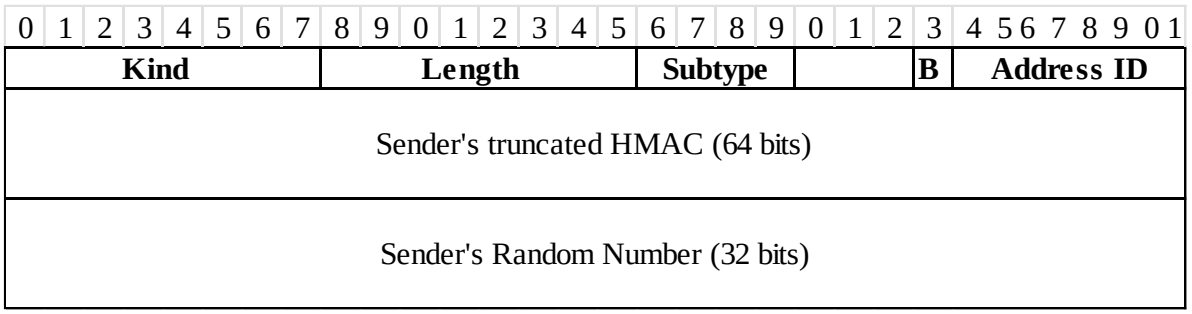

Figure 2.10 Join Connection (MP_JOIN) Option (for Responding SYN/ACK) **[3]**

Three of the MP_JOIN option's flag bits in SYN packets are currently set aside and their value required to be zero during transmission. The final “B” bit specifies whether the sender wants this sub-flow only at the time when no other routes are available for communication (B=1) or if it wants to use the path for data transmission instantly (B=0). Sender requests the other host to only sends data on this sub-flow by setting B=1 when there are no other sub-flows available for which value of B sets to 0 [3].

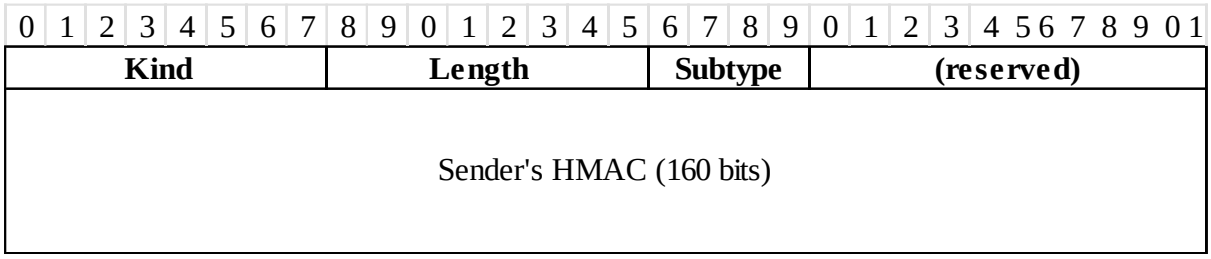

Figure 2.11 Join Connection (MP_JOIN) Option (for Third ACK) **[3]**

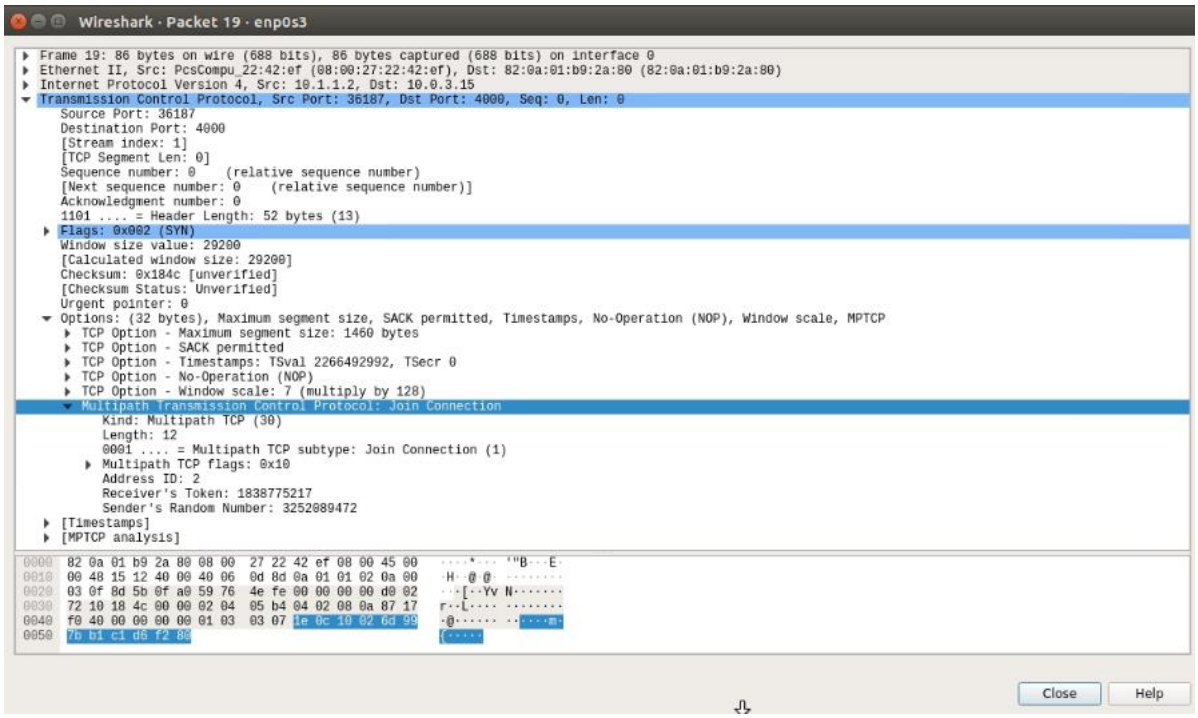

Figure 2.12 Wireshark Capture of MPTCP MP_JOIN packet

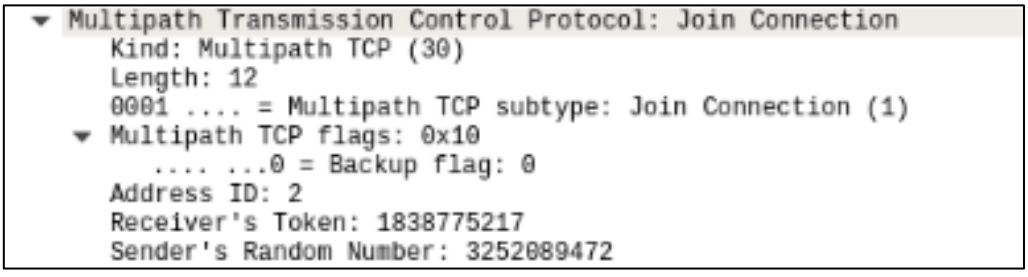

Figure 2.13 MP_JOIN option in MPTCP SYN

After verifying the token received from the Alice, Bob will transmit the SYN/ACK + MP_JOIN packet to the Alice. This packet also contains random number and truncated HMAC as shown in packet header format in Figure 2.10 if token value obtained is accurate otherwise the connection will be reset if the token is invalid. The use of 32-bit token prevents blind state exhaustion attacks. The procedures will not permit a host to function stateless with MP_JOIN stage, even though cryptographic operations are required for authentication. Following receipt of the SYN/ACK, Alice will one more time reply with an ACK packet that includes the MP_JOIN option and the HMAC value of Alice's key $K_{Alice.}$ Only the first 64 bits of the HMAC values were sent because there was not enough room for them in the TCP header.

As shown in Figure 2.11, the initiator includes its authentication data in the ACK (the third packet from the packet sequence). Due to the unique nature of this HMAC, its transmission must be guaranteed at all costs. Hence, a standard TCP ACK must be returned upon receipt, and the packet required to be resent in case of the ACK is not delivered. The sub-flow remains in the PRE-ESTABLISHED state until the recipient acknowledges the ACK/MP JOIN packet. While in the PRE-ESTABLISHED stage, data transmission is prohibited. The sender is required to set all reserved bits in this option to 0. Hence, Bob will then send an ACK packet to complete the connection. Following that, Alice and Bob can communicate via both interfaces. Figure 2.8 depicts the MP_JOIN procedure [22]. Figure 2.12 and 2.13 shows the Wireshark capture of the SYN+ MP_JOIN packet of MPTCP.

2.1.3 ADVERTISEMENT OF THE NEW ADDRESS TO ANOTHER HOST

Any host may announce the availability of additional addresses anytime throughout the lifetime of MPTCP connection. There are two methods for promoting the additional address. One of them broadcasts an accessible address with the ADD_ADDR option (Explicitly), while the other uses the MP_JOIN option to initiate a new sub-flow (Implicitly) [27]. The ADD_ADDR option also comprises the Address ID to map the network address with a unique number. Every address is mapped with the unique Address ID. When address translators are in play, this is how identical MP_JOIN options are located. In order to delete an address from a connection, you'll need to know its unique Address ID. The main purpose of Address ID is to make the sender capable of identifying the connection uniquely, although the technique used for generating Address ID is implementation dependent. As shown in Figure 2.14, only single packet is required to advertise the new address explicitly which doesn’t contain any

authentication details, so it can be easily forged to initiate an attack. Figure 2.16 and 2.17 depicts the Wireshark capture of the ADD_ADDR packet.

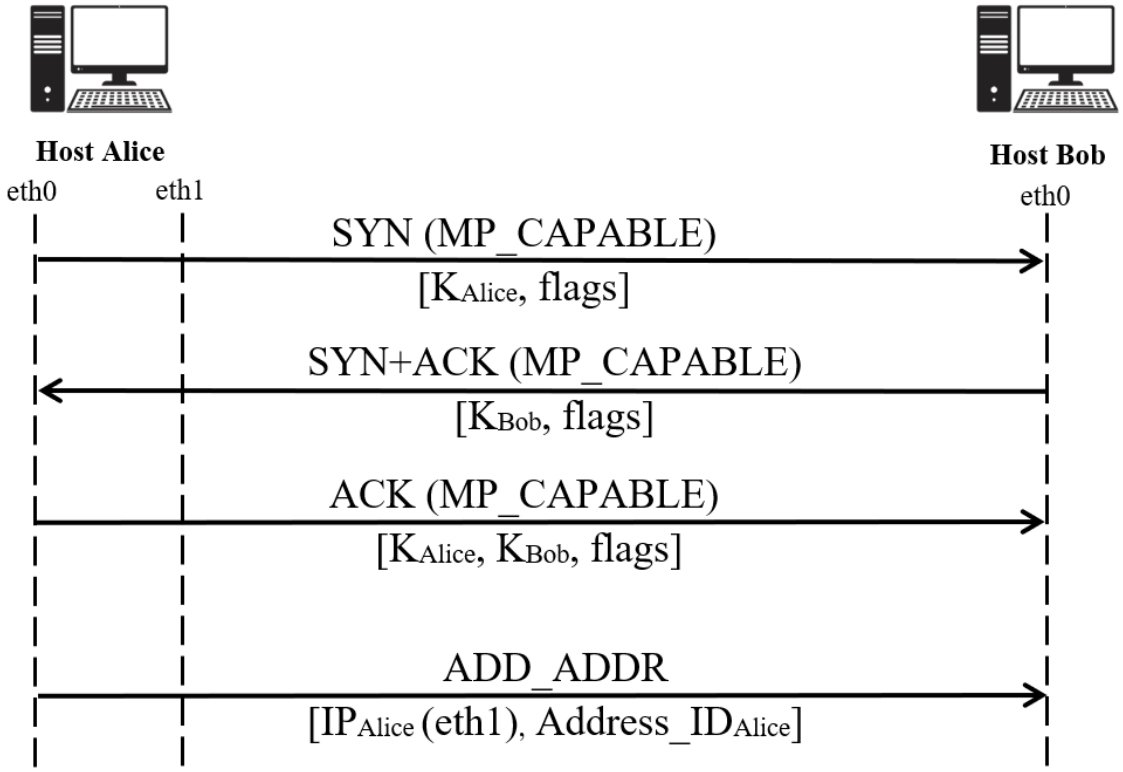

Figure 2.14 Advertisement of new address using ADD_ADDR packet

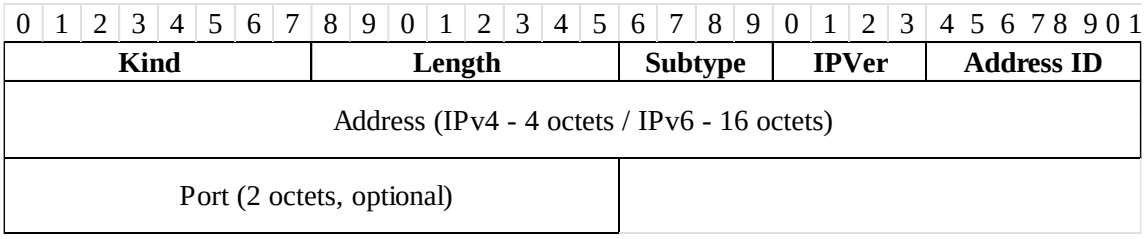

Figure 2.15 ADD_ADDR MPTCP header format

The last two octets depict the TCP port as shown in Figure 2.15 is optional. In majority cases, the same port will be used for future sub-flows which is used for initial sub-flow but in special cases, host can specify the port number through this field. MPTCP uses the same port on which ADD_ADDR packet was sent if port is not specified explicitly.

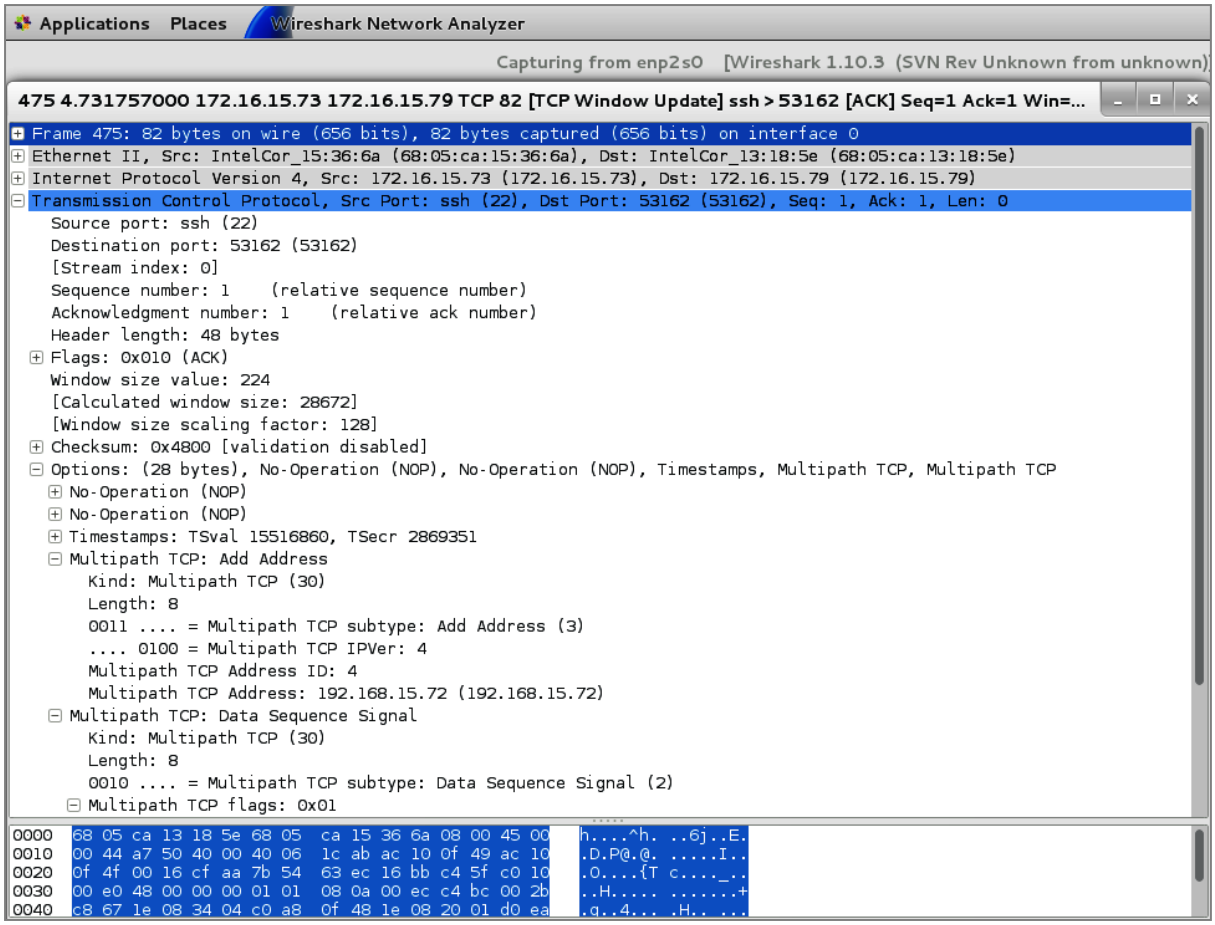

Figure 2.16 Wireshark Capture of ADD_ADDR packet

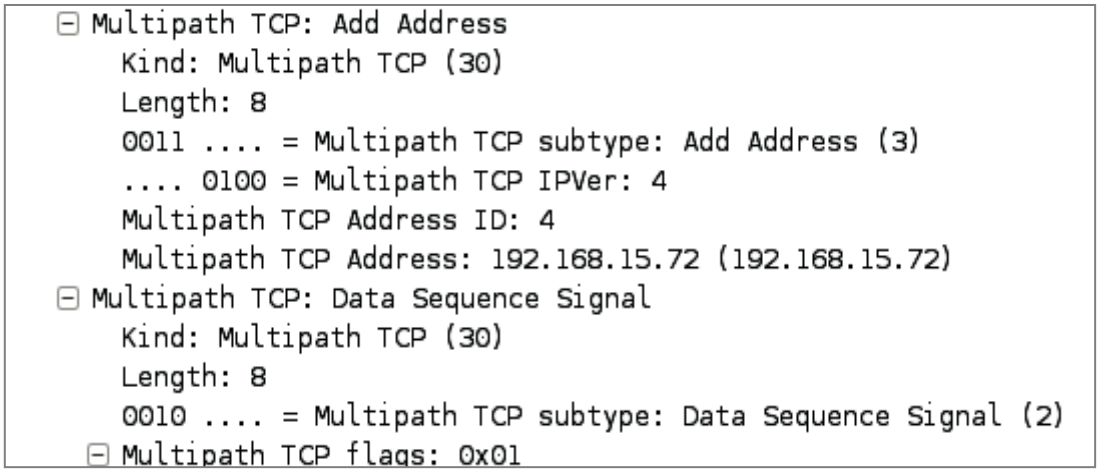

Figure 2.17 ADD_ADDR Option

2.1.4 REMOVING THE IP ADDRESS

REMOVE_ADDR is another option of MPTCP which can be used to remove any of the announced addresses that need to be [3]. All connections using that address will then be terminated when it announces the address ID of the specific interface. Throughout the lifespan of an MPTCP connection, if an advertised address got compromised or unavailable due to any

reasons, the impacted host should make the peer aware of this so that it can delete any sub-flows associated with the faulty address.

2.1.5 DATA SEQUENCE SIGNAL (DSS)

The focus of the Data Sequence Signal (DSS) is on the information being transmitted. At the MPTCP level, the Data Sequence Number (DSN) is used to identify each data byte, just as it is in TCP [3]. MPTCP connection is assigned a 64-bit unique sequence number, DSN to guarantee reliable and in-order delivery of packets through sub-flows. This value is initially generated randomly and then incremented by one for each byte of transmitted data. A "Data Sequence Signal" is transmitted with the "Data Sequence Mapping" embedded in it. The length, sub-flow sequence number (SSN), and DSN are part of the data sequence mapping. An acknowledgment of the received DSN at the connection level ("Data ACK") is also possible with this channel [3].

Every MPTCP sub-flow advertises the same size and location of it receive window and uses the same receive buffer. In MPTCP, there are two forms of recognition. Regardless of the DSN, standard TCP acknowledgments are utilized on each sub-flow to confirm the successful delivery of data packets transmitted via the sub-flow. In addition, the data sequence space is acknowledged at the connection level. These confirmations follow the progression of the byte stream and shift the receiving window accordingly [3].

The data delivered on a particular sub-flow may (or may occasionally be required to) be sent to other sub-flows if the sub-flow is closed, has congestion, or even experiences network failures. It's known as reinjection. The DSS option has to be adjusted (at least the SSN is modified). Segments sent once more on the same sub-flow are referred to as re-transmissions (this is simply the TCP re-transmission, in case of packet losses).

2.1.6 THE ABRUPT CONNECTION RELEASE (MPTCP FAST CLOSE)

MP_FASTCLOSE (The Fast Close) option of MPTCP works similarly to the TCP RST option, which is an abrupt connection release. To terminate the MPTCP connection, the terminating host must send an ACK segment with MP_FASTCLOSE option on one sub-flow and RST segments on all other sub-flows. The other host then closes the sub-flow with the RST segment, confirming receipt of the MP_FASTCLOSE option. Now, the entire MPTCP connection can be disconnected. Once the other host has transmitted the RST segment for the final sub-flow, the host can terminate the connection. After the first retransmission timeout

elapses without receiving the RST segment, the MP_FASTCLOSE should be resent. The authentication key of the remote host is stored in the MP_FASTCLOSE option to ensure the authenticity of the data. Since we no longer require the key, it is transmitted unencrypted [3].

2.1.7 CHANGE IN MPTCP SUB-FLOW PRIORITY

At the time of initial sub-flow configuration, hosts can specify whether they want the sub-flow to function as a primary or backup path. Using the MP_PRIO signal, Host Alice can ask Host Bob to adjust the priority of a sub-flow while the connection is active. MP_PRIO option can be used to update the backup flag (B flag) of the sub-flow. The sub-flow should be utilized only when no other sub-flows are available (although local policies or technical difficulties may override this flag) with the B flag set to 0 [3].

2.2 CRYTOGRAPHY AND NETWORK SECURITY

This section briefly introduces about cryptography and network security. The various terminologies used with cryptography, basic security attacks and security goals are discussed in this chapter. Moreover, the basics of Elliptic Curve Cryptography (ECC) and Identity Based Encryption (IBE) are covered which are used in proposed model.

2.2.1 CRYPTOGRAPHY, SECURITY ATTACKS & SECURITY GOALS

Cryptography is the science behind offering the protection of data. Cryptographic methods are not limited to protecting the transmission of information; data held on servers is also commonly protected [34] [35]. It demands that a certain secret be known only by the legitimate audience. To offer confidential communication, mutually trusted parties have used cryptography for thousands of years. Figure 2.18 shows the conventional network security model for information exchange among two hosts. Over some network, a message is to be transmitted from one location to another in which two principals (Sender and Receiver) involved must work together. Two parties use communication protocols (such TCP/IP) and predefined path over internet among them to establish a logical information channel. Security aspects need to be considered at the time of requirement of protection against security threats. It is required to consider major two aspects (i) security related information (ex: encryption) and (ii) secret information (ex: key) at the time of using any security technique.

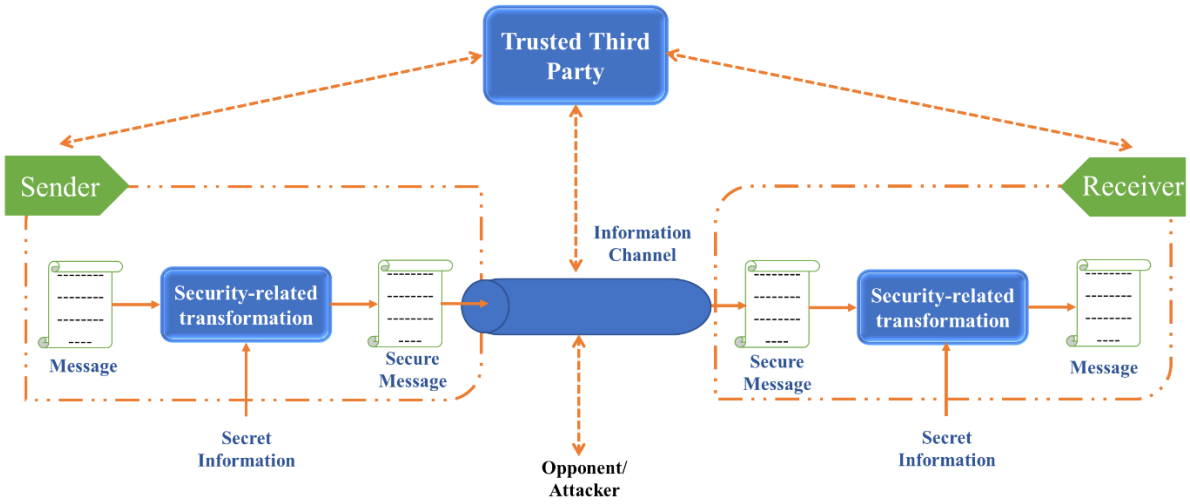

Figure 2.18 Network Security Model

The mutually trusted parties can use the secret key, decided before the communication, to convert the original message, also known as the plaintext, into a scrambled form that is incomprehensible to anyone who does not have the key. The message scrambled throughout this procedure is known as the ciphertext, which can be converted to the original message using decryption. According to Kerckhoffs' approach [35], the key (a password, pin, etc.) should be kept secret, not the algorithm/technique used. It permits the creation of robust public algorithms that have been reviewed and accepted by a large audience.

The fundamental security goals are as below [34]:

A. Confidentiality:

Confidentiality is to hide the information or message from unauthorized parties. No other person other than the receiver can understand the information or message the sender provides. It is one of the security objectives which can be accomplished by applying distinctive encryption methods.

B. Integrity:

Integrity ensures that the information the sender sends is not modified or altered by any third party while transmitting the information or at storage. It is a security goal that can be achieved by applying various encryption techniques.

C. Availability:

Availability is an important aspect of cryptography, as the data or information should be available whenever required or requested. Only data storage, confidentiality, and integrity will not work if we will not get any data or information while it is needed.

Based on threatening the security goals, the major security attacks [34] [36]can be categorized as follow:

A. Security attacks threatening Confidentiality

a) Snooping: It can be defined as an unauthorized access or interception of data being communicated over the channel.

b) Traffic Analysis: Despite the apparent regularity of the message traffic between sender and receiver, neither party is aware that the messages have been read or that a third party has observed the traffic pattern.

B. Security attacks threatening Integrity

a) Modification: Some part of a valid message is manipulated, or messages are either delayed or reordered without the user's knowledge.

b) Masquerading occurs when one entity pretends to be another to gain an advantage.

c) Replaying: A data unit is secretly taken and then resent to achieve an unlawful result through this process, which incorporates passive data capture.

d) **Repudiation:** The sender or the recipient can carry out this assault. Either the sender or the recipient can claim.

C. Security attacks threatening Availability

a) Denial of Service (DoS): DoS attacks are launched to obstruct legitimate users' access to online services and resources.

The above-mentioned attacks can be further classified among two categories on the basis of their impact on data, service or resources.

A. Active Attack: An active attack is a form of network exploit in which the attackers actively change the data or makes conscious efforts to corrupt or otherwise compromise system resources. The victims will suffer consequences as a result. Before launching an aggressive attack, the attackers will conduct passive attacks to gather intelligence. The attackers attempt to cause disruptions and break security. Their security and accessibility will be at risk from such an assault. Attacks that need active participation are more challenging to execute.

Example: Modification, Masquerading, DoS attack, Replaying, Repudiation

B. Passive Attack: In a passive attack, the attacker merely keeps tabs on the system or makes use of the data it collects. The purpose of adversary in a passive attack is to gather intelligence without actively disrupting the target system. There will be no change to the system's resources or data as a result of the attack. Typically, both the sender and the recipient of the messages have no idea about the attack, as the messages are transmitted and received in an apparently regular manner. Since passive attacks are conducted invisibly, it is challenging for the victim to detect their occurrence. The objective of a passive assault is to collect data or to search for weak spots in a network.

Example: Snooping, Traffic Analysis, Eavesdropper, etc.

The basic terminologies used with cryptography are as follow [37] [38]:

A. Plain Text (P.T.): Plain Text is a text humans can understand through reading. It is also known as the original message or normal text, or clear text. It is a kind of message which the sender sends to the receiver via an encryption process.

B. Cipher Text (C.T.): Cipher Text is a text humans cannot understand as it is in an unreadable or meaningless form. It is and of message received by a receiver after the encryption and before the decryption process. For secure communication between sender and receiver, P.T. is translated into the C.T.

C. Encryption: A process that is used to generate C.T. from the given P.T. to provide security to the original message is known as encryption. A key and encryption algorithm are required for the encryption process. This process takes place on the sender's side.

D. Decryption: A process that is the inverse of encryption is called decryption. It is a process of converting the C.T. into the P.T. to make it recognizable text for the receiver. A key and algorithm are required for the decryption process. This process takes place on the receiver side.

E. Key: The key is the heart of cryptography. It is one of the aspects of providing security to information while communicating via insecure channels. The key is used on both the sender and receiver sides. It can be anything – a number, a function, or an algorithm [39]. On the sender side, it is used in the encryption process to hide the data (message); on the receiver side, it is used in the decryption process to retrieve the data (message).

F. **Key Size:** It is the length of the key. It can be in bits or bytes.

G. **Block Size:** When a sender sends very large data to the receiver, it divides the whole data into small blocks of fixed size. This fixed string length in a single block is the block size. It depends on the encryption algorithms.

H. **Round:** Confusion is created by adding multiple rounds of the encryption function in the encryption process to generate cipher text to hide messages more securely and efficiently. The number of rounds depends on the encryption algorithm.

I. **Cryptanalysis:** Cryptanalysis is a process to find the solution to decrypt the encrypted message with the help of mathematical formulas used to search the vulnerabilities of algorithms.

The below security mechanisms can be used to achieve the security goals [37] [40]:

A. **Authentication:** Authentication ensures that the sender and receiver claim their identity, confirming the origin and destination of the message or data.

B. **Access Control:** This security service makes sure that a person who has the authority to access the data, only that person can access that particular data to meet the security goals.

C. **Non-repudiation:** This security service provides a certainty that the sender or receiver will not deny the action done by them on the information or data later on.

D. **Digital Signature:** This process ensures that the data or information received at the receiver side is the same as the data or information sent by the sender. It fulfills both the aspect – Authentication and Integrity. As the digital signature is added to the data or information, the receiver can identify that the message is from an authorized person. Any other entity changes the data if the digital signature does not match.

The most well-known security element, confidentiality, is offered through encryption and decryption. Asymmetric and symmetric ciphers are the two groups into which these techniques can be categorized (with public and private keys) [41].

Data integrity (and authenticity), the next security feature, can be provided through the Message Authentication Code (MAC), a token derived from messages and calculated with

symmetric keys. The MAC nevertheless ensures the message's author knows the correct secret key, even if it cannot guarantee the author's true identity.

Cryptography also improves data availability by guaranteeing that only authorized users can gain access to systems and retrieve data in a reliable and timely way. It ensures that information systems are available and reliable at all times. Linking virtual identities, i.e., holders of private, asymmetric keys, etc., to actual identities, certificates, and formal digital documents; fills the gap.

Information security helps various industries to compete in today's modern era. Cryptography is one of the solutions which can be used to get protection of proprietary data against various threats by utilizing it in appropriate way. As mentioned, one way to classify cryptosystems is by the number of keys they employ, public and private key cryptography. One of the most effective methods of ensuring data confidentiality is through encryption by the data producer.

The same keys are used by symmetric key cryptosystems to cipher and decipher the data as shown in Figure 2.19. Therefore, the recipients of the data must be aware of the keys. The primary problem with symmetric cryptography is safely swapping the keys among communicating parties before communication begins.

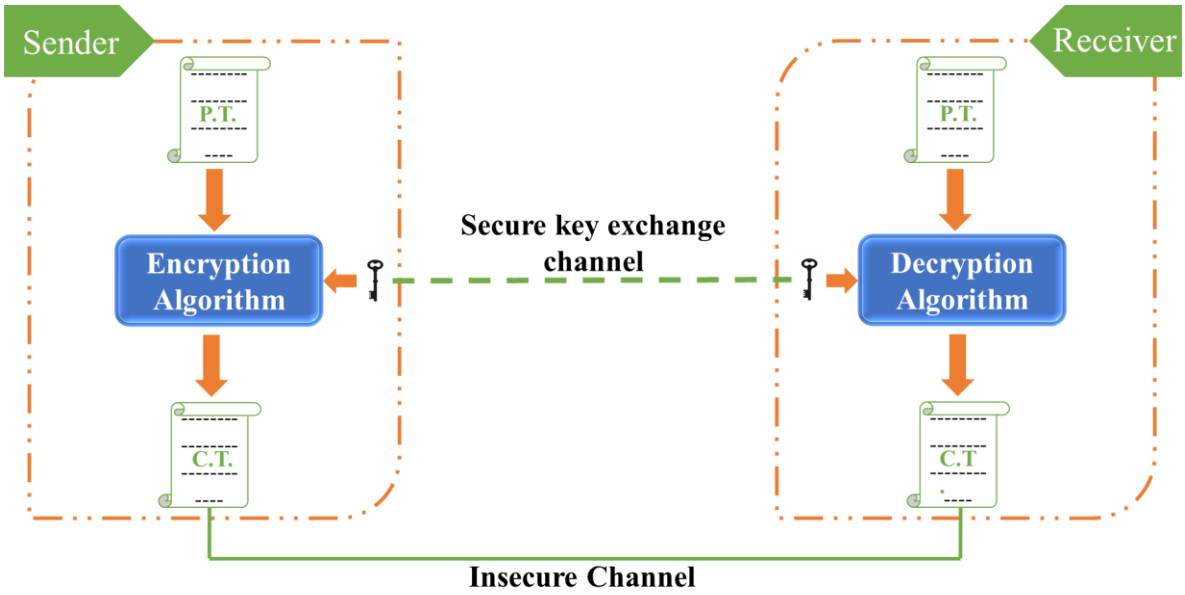

Figure 2.19 Private Key Encryption (Symmetric Key Encryption)

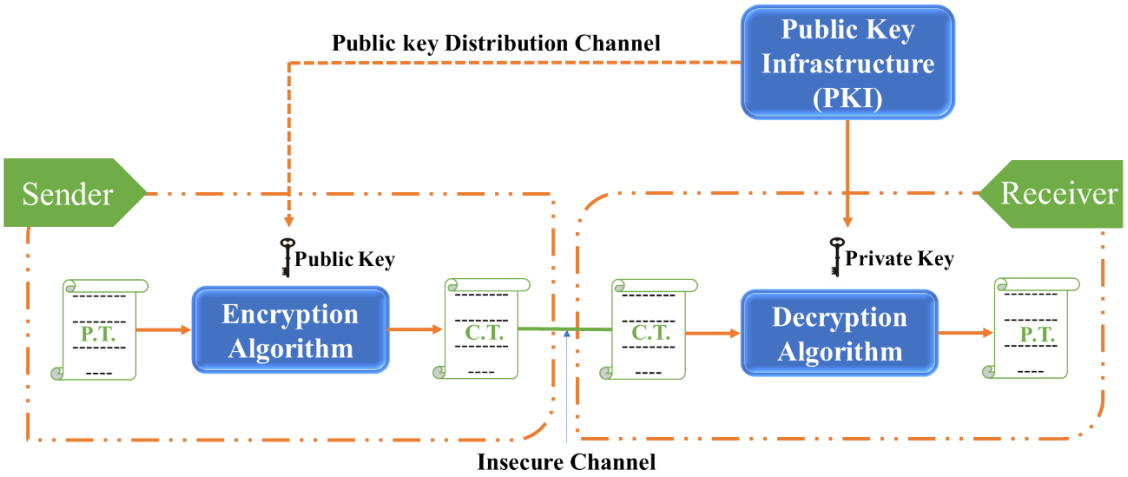

Figure 2.20 Public Key Encryption (Asymmetric Key Encryption)

Asymmetric key cryptosystems, or public key cryptography, employ public and private keys for encryption and decryption as shown in Figure 2.20. As their names imply, only the private key used for decryption must be kept secret by its owner; the public key, however, can be delivered clearly without any security considerations. No prior key exchange is required, in contrast to symmetric ciphers. However, it increases the computational cost (100-1000 times less efficient). Thus, only tiny data blocks, generally transmitting a symmetric encryption key, are sent using asymmetric key ciphers.

There are several robust and effective conventional encryption and hashing methods like Data Encryption Standard (DES), Triple DES, Advanced Encryption Standard (AES), Blowfish, International Data Encryption (IDEA), Rivest–Shamir–Adleman (RSA), Elliptic Curve Cryptography (ECC), Secure Hash Algorithm (SHA)-512, Message Digest (MD)5, and many more. These algorithms only offer confidentiality and/or integrity but are not suitable for implementing granular security measures like access controls.

2.2.2 ELLIPTIC CURVE CRYPTOGRAPHY (ECC)

The ECC [28] is based on the elliptic curve $E_p(a, b)$ over finite field $GF(2^m)$, which satisfies the below cubic equation.

$$y^2 = x^3 + ax + b, \text{ Where } 4a^3 + 27b^2 \neq 0, a, b, x, y \in GF(2^m)$$

Here x *and* y are the variables, and a *and* b are coefficients. The equation represents a non-singular elliptic curve where $x^3 + ax + b = 0$ has three distinct roots. Based on the selection of a *and* b, the shape of the curve may vary. Figure 2.21 shows the curve for $y^2 = x^3 - 4x + 5$.

The discrete Logarithmic Problem (DLP) and the Diffie-Hellman problem (DHP) define the ECC. The Elliptic Curve Diffie-Hellman Protocol (ECDHP) is a key agreement protocol based on the Diffie-Hellman algorithm applied to an elliptic curve. In this context, ECDLP extends DLP over a finite field [34] [36].

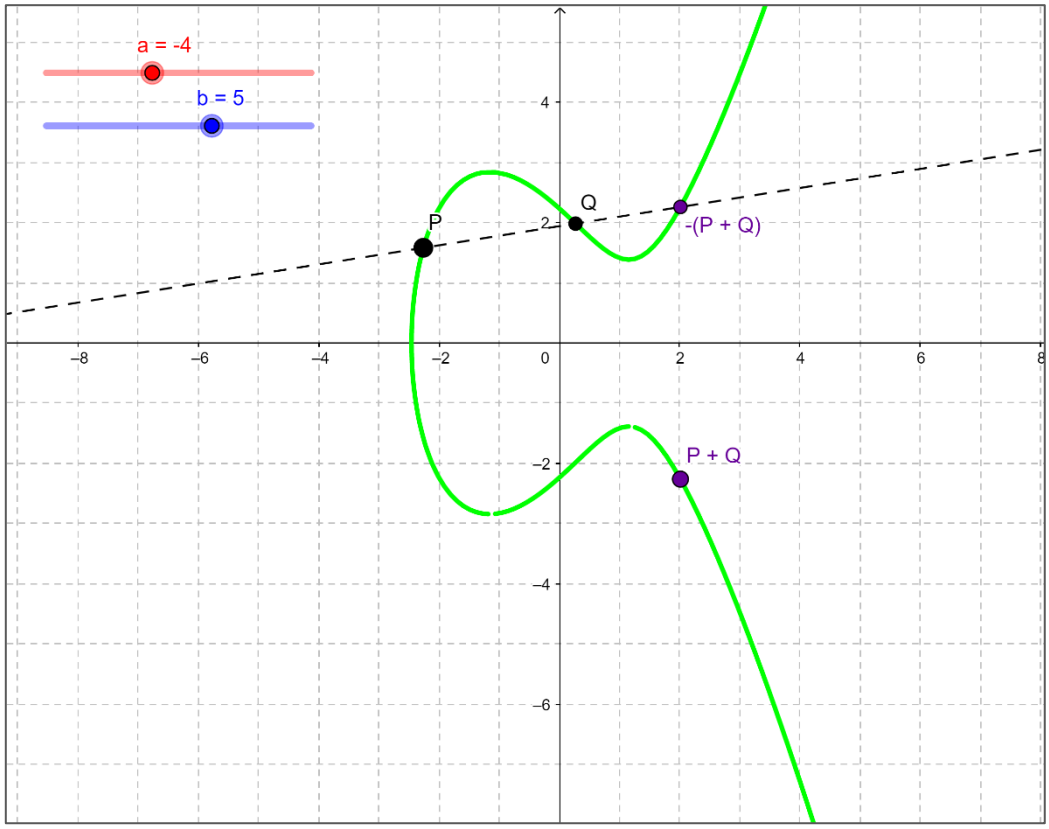

Figure 2.21 Elliptic curve for $y^2 = x^3 + ax + b$, where a=-4 and b=5

ECDLP can be defined as considering the equation $Q = kP$, the discrete logarithm of Q to the base P, for the given points P and Q in the groupE_p (a, b) and $k < p$. It will be extremely difficult for an adversary to determine the value of k.

ECDHP uses elliptic curves for key exchange. First, select the large integer number q. Here, q must be selected in such a way that it is either the prime number p for the Z_p or an integer of the form 2^m. Also, select the coefficients a and b for the elliptic curve $E_q(a, b)$. Now select the point $G = (x_1, y_1)$ On the elliptic curve of the order of large number n. Here, $E_q(a, b)$ and G are the global parameters that will be used for the key generation and will be known to all the participants.

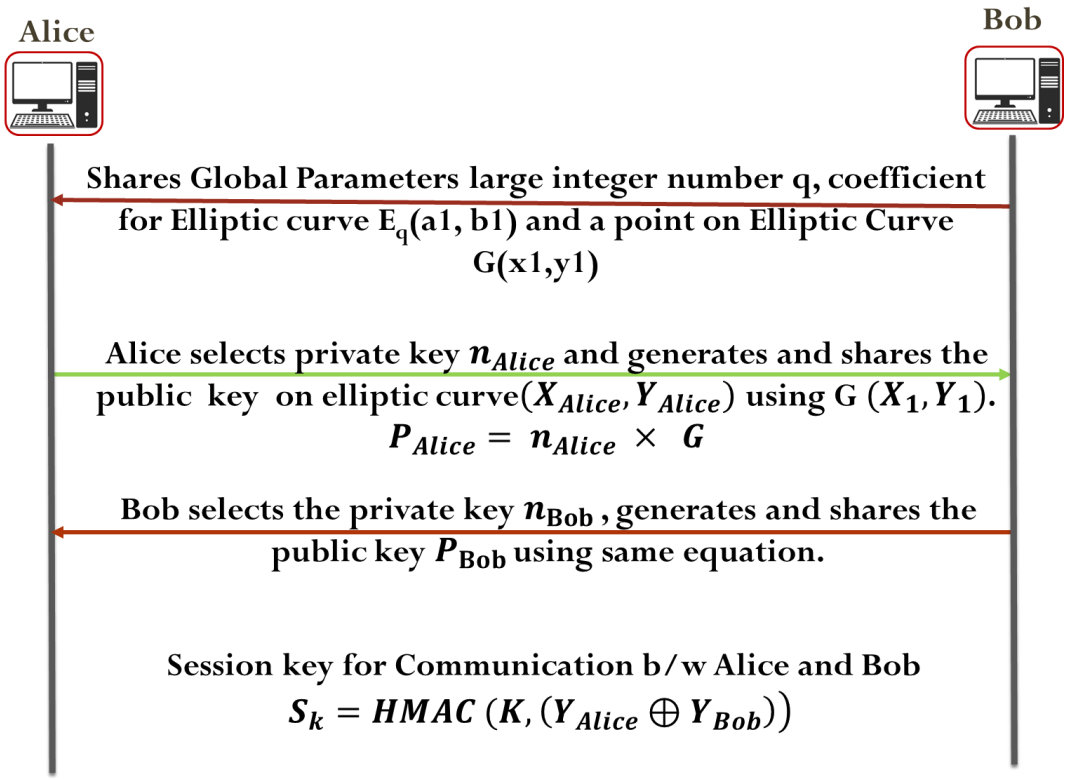

Figure 2.22 Key Exchange using ECC **[28]**

The key exchange process using communicating hosts like Alice and Bob is as follows [28]:

1. Alice selects the private key n_{Alice} Which is less than n. The public key P_{Alice} will be the point (X_{Alice}, Y_{Alice}) in elliptic curve $E_q(a, b)$ which can be generated by using the private key n_{Alice} And global parameter G using the below equation.

$$Public\ Key\ P_{Alice} = n_{Alice} \times G$$

2. By using the same equation, Bob can select the private key n_{Bob} And generate the public key $P_{Bob.}$

3. Now, Alice can generate the secret key $K = n_{Alice} \times P_{Bob}$ And Bob can generate the secret key $K = n_{Bob} \times P_{Alice}$. Here, the secret key generated by Alice and Bob will be the same, which will be used for communication.

Session Key for communication between Alice and Bob is as below:

$$S_k = HMAC\ (K, (Y_{Alice} \oplus Y_{Bob}))$$

The whole process is shown in Figure 2.22 [28]

Transport layer opportunistic security techniques like tcpcrypt use asymmetric protocols like Rivest-Shamir-Adleman (RSA) and (Diffie-Hellman) DH, necessitating a lengthy key for

secure key exchange. There is also the issue of the MitM attack, which is detrimental to the DH approach-based approaches. When exchanging a long security key between communicating hosts, a lot of extra space in the packet header is needed because of the computation required to compute the security parameters. Although MPTCP services are activated in the TCP header's options field, this space is not exclusively dedicated to MPTCP options and only has a capacity limit of 40 bytes. Consequently, the current MPTCP relies on a secret key length of 64 bits, which is inadequate.

Table 2.2 Security level comparison of RSA and ECC **[42] [43]**

Security bit level	RSA (Key Size)	ECC (Key Size)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Compared to RSA, the ECC's key size is less while maintaining the same level of security as shown in Table 2.2, which is suitable for use in the proposed system.

2.2.3 IDENTITY-BASED ENCRYPTION (IBE)

To streamline the process of issuing and managing certificates for electronic communication, Shamir suggested Identity-Based Encryption (IBE) in 1984 [44]. The algorithm simplifies the public key infrastructure by allowing any string to be used as a public key for encrypting the message. Assume that Alice, the sender, encrypts an email and sends it to Bob, "Bob@mail.com," with the help of Bob's public key. After receiving Alice's encrypted message, Bob authenticates with a private key generator (PKG) to obtain the private key necessary to decrypt the message. Bob will receive the private key in his mailbox. It enables the sharing of encrypted mail without creating a public key infrastructure [45].

The IBE consists of 4 functions for executing the complete scenario. **SetUp** produces system parameters and a master key. **KeyGeneration** function produces the private key, which belongs to the public key, with the use of the master key and the system parameters, **Encryption** function uses for the encryption of the message using any arbitrary string, and the **Decryption** function uses for the decryption of any message [44].

I. **SetUp:** Using this algorithm, the PKG provides the system parameters and master-key share, which will be used to generate the private key. This algorithm takes security parameter k $\in Z^+$, G1 and G2 to build a bilinear map $\hat{e} : G1 \times G2 \rightarrow G2$, 2 cryptographic hash functions $h1$ and $h2$ and returns system parameters $Params = \langle q,\ G_1,\ G_2, \hat{e}, n, P,\ P_{pub}, \text{H1}, \text{H2} \rangle$ and master key share. Here, q is the prime number, and P is the random number generator, which generates the random number s to obtain the value of $P_{pub} = sP$, where s is the master key share.

II. **KeyGeneration:** By using this algorithm, the private key of the particular entity will be generated on request by using system parameters, master-key share, and public parameters. This algorithm uses the master key share and ID to generate the private key $d_{ID} = sQ_{ID}$, where s is a master key and $Q_{ID} = H1(ID) \in G_1^*$.

III. **Encryption:** During the encryption, the ID of the receiver and system parameters will be used to encrypt a message, and ciphertext will be generated. This algorithm computes the Cipher Text C using $\langle rP, \sigma \oplus H2\ ({g_{ID}}^r) \rangle$, and $g_{ID} = \hat{e}(Q_{ID}, P_{pub}) \in G_2$, σ is a random number and $r = H3(\sigma, M)$.

IV. **Decryption:** The private key generated by the key generation algorithm, system parameters generated by the SetUP algorithm, and the receiver ID will be used to decrypt the ciphertext. The Plaintext message M can be computed as below: if $C = \langle U, V, W \rangle$ is a cipher text encrypted using public key ID, compute $V \oplus H2\ (\hat{e}\ (d_{ID}, U)) = \sigma$ and $W \oplus H4\ (\sigma) = M$ and $r = H3(\sigma, M)$. Test that $U = rP$. If not, reject the cipher text. Here, M is the decryption of C.

The below mentioned steps illustrates the process of Identity Based Encryption:

1. Sender generates the system parameters by using the SetUP function.
2. Sender encrypts the message using Encryption function by passing any arbitrary string/ID of receiver as an encryption key.
3. After receiving and encrypted message from the sender, receiver will call the SetUP function to generate the system parameters and request to the PKG for the master key by providing public key (arbitrary string/ID).
4. PKG will generate the master key for the received ID using the KeyGeneration function and sends the generated key to the receiver.

5. Receiver will get the private key corresponding to the public key with the use of the key provided by PKG and system parameters and get the original message using Decryption function.

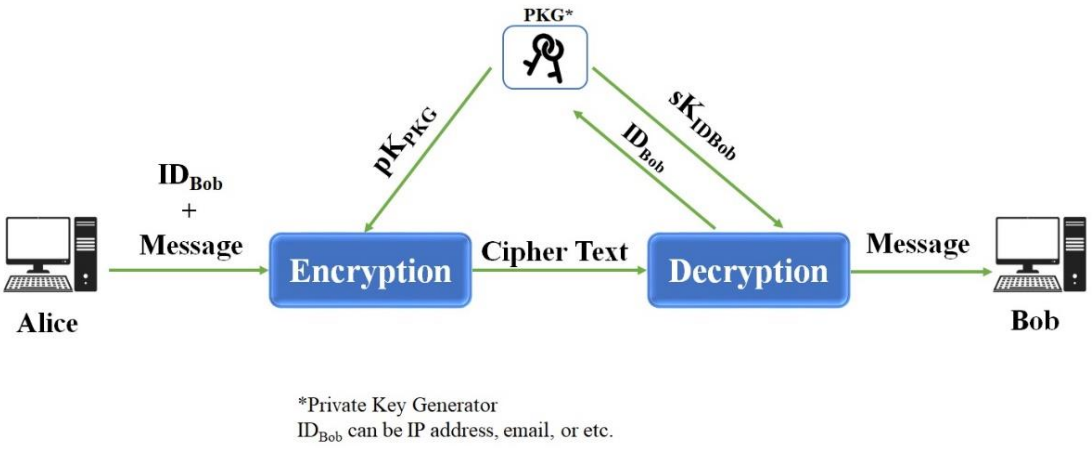

Figure 2.23 Identity-Based Encryption

As shown in Figure 2.23 and from the steps mentioned above, it can be observed that sender doesn't require to perform any tasks to encrypt the message. It can use any arbitrary string of receiver such as email id, IP address, etc. as a public key for the encryption which makes encryption process computationally easy. PKG produces the private key corresponding to the arbitrary string for the receiver. Receiver will generate the private key from the master key provided by the PKG for the decryption process.

When compared to the standard PKI system, the IBE scheme has many advantages such as less complexity of encryption process, no certificate required, etc. IBE's ability to encrypt messages without a public key infrastructure (PKI) is a key feature. In this context, an "identity" is a "string of characters." The lack of error-tolerance in the identity-based encryption strategy is a fundamental drawback of this method. Because of the inherent noise in biometric measurements, user biometrics like iris scans, fingerprints, etc. cannot be utilized as identities for encryption. A Fuzzy Identity Based Encryption technique can be used to accommodate both fault tolerance and biometric identifiers. Moreover, the private key generator and user must communicate over a safe channel. PKG requires a robust level of security because it stores all private keys and must always be accessible via the network. There is no preemptive data sharing because encrypted data can only be decoded by a single trusted user.