4. PROPOSED SYSTEM ARCHITECTURE AND METHODOLOGY

The current version of MPTCP has many security flaws like ADD_ADDR vulnerability, SYN + MP_JOIN vulnerability, eavesdropper in the initial handshake, etc. These vulnerabilities can be exploited to launch dangerous security attacks like MitM attack, DoS attack, and session hijacking attack, which puts data confidentiality, integrity, and availability over the connection. The proposed model is to exchange the keys securely to protect MPTCP communication from ADDR_ADDR attacks and eavesdroppers during the initial handshake, which can be used in the future to authenticate the sub-flow.

4.1 SECURE KEY EXCHANGE MODEL FOR MPTCP (SKEXMTCP) USING IDENTITY-BASED ENCRYPTION

The Secure Key Exchange Model for MPTCP (SKEXMTCP) using Identity-Based Encryption is proposed to offer secure communication of keys. When exchanging the security parameters for MPTCP during the initial handshake, the SKEXMTCP uses the identity-based encryption approach to share secret keys. The proposed model will offer protection against eavesdroppers during the initial handshake, which prevents an ADD_ADDR attack in the process.

IBE employs a Private Key Generator (PKG), a third-party authority, which distributes the private keys to the communicating hosts by their identities (such as their email addresses, IP addresses, etc.). Here, the sender and receiver's private keys will be generated using the IP address and port of the communicating host as a public parameter. The suggested remedy consists of two modules: (i) Private key generation (SKG_SKEXMTCP) and (ii) the initial handshake, which uses key pairs to exchange session keys. Table 4.1 shows the terminologies used in the proposed model.

Table 4.1 Terminologies used in the proposed model

Term	Significance/ Meaning	Generation
PU_{Alice}	The public key of Alice	Alice's IP address and port combination will be used as a Public Key.
PR_{Alice_Master}	The shared key used to generate the private key of Alice	Generated by PKG and shared with Alice.
PR_{Alice}	Private Key of Alice	It can be generated by using PR_{Alice_Master} and PU_{Alice}.
PU_{Bob}	The public key of Bob	Bob's IP address of the Bob will be used as a Public Key.
PR_{Bob_master}	The shared key used to generate the private key of Bob	The key will be generated by PKG and shared with Alice.
PR_{Bob}	Private Key of Bob	It can be generated by using PR_{Bob_Master} and PU_{Bob}.
ID_{Alice}	ID used as a public key of Alice	IP_{Alice} + $Port_{Alice}$ Combination.
ID_{Bob}	ID used as a public key of Bob	IP_{Bob} + $Port_{Bob}$ Combination.
Pub_Param	Public Parameters	Public parameters are generated by PKG which will be shared with hosts. These parameters will be used to generate private key of individual host from master key.
request(S, D, P)	Request function	This function sends request from S to D with P parameter.
response (S, D)	Response function	This function sends response from S to D which returns public parameters and master key.
Key_Gen(P, M, ID)	Key generation function	Generates private key from public parameter, Master key and ID of host.

4.1.1 PRIVATE KEY GENERATION (SKG_SKEXMTCP)

As shown in Figure 4.1, In the Key Generation (SKG_SKEXMPTCP) module, the IP addresses of Alice and Bob, respectively, IP_{Alice} and $IP_{Bob,}$ will be used as public parameters by PKG to produce private keys for communicating hosts Alice and Bob over MPTCP. These private keys will encrypt the initial handshake data in the MPC_SKEXMTCP module. The PKG will use the host's digitally signed IP address and port combination to verify the authenticity of the communicating host.

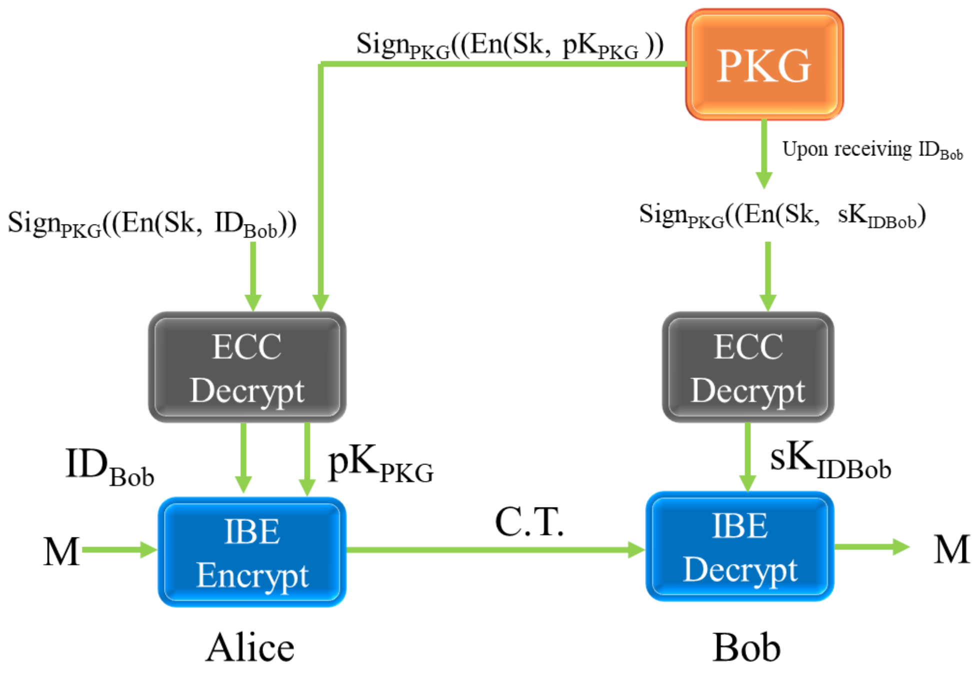
Sign$_{PKG}$((En(Sk, pK$_{PKG}$))
PKG
Upon receiving ID$_{Bob}$
Sign$_{PKG}$((En(Sk, ID$_{Bob}$))
Sign$_{PKG}$((En(Sk, sK$_{IDBob}$)
ECC Decrypt
ECC Decrypt
ID$_{Bob}$
pK$_{PKG}$
sK$_{IDBob}$
M
IBE Encrypt
C.T.
IBE Decrypt
M
Alice
Bob

Figure 4.1 Key Generation with IBE

4.1.1.1 STEP 1. HOST ALICE KEY GENERATION

a) Here, the public key of Alice PU$_{Alice}$= ID$_{Alice}$. The sender can use it to encrypt the messages for Alice.

b) Host Alice requests PKG with ID$_{Alice}$ (IP$_{Alice}$ and Port$_{Alice}$) as a parameter by authenticating itself using a digitally signed IP address and port combination.

c) PKG calculates the share of Alice PR$_{Alice\_master}$, and it will be sent back to Alice.

d) Alice can calculate the private key PR$_{Alice}$ from the PR$_{Alice\_master}$ Public Parameters and ID$_{Alice}$. PR$_{Alice}$= Key_Gen(Pub_param, PR$_{Alice\_master}$, ID$_{Alice}$). The messages encrypted by PU$_{Alice}$ can be decrypted using PR$_{Alice}$.

4.1.1.2 STEP 2. HOST BOB KEY GENERATION

a) Here, the public key of Bob PU$_{Bob}$= ID$_{Bob}$. The sender can use it to encrypt the messages for Bob.

b) Host Bob sends a request to PKG with ID$_{Bob}$ (IP$_{Bob}$ and Port$_{Bob}$) as a parameter by authenticating itself using a digitally signed IP address and port combination.

c) PKG calculates the share of Bob PR$_{Bob\_master}$, and it will be sent back to Bob.

d) Bob can calculate private key PR$_{Bob}$ from the PR$_{Bob\_master}$, Public Parameters and ID$_{Bob}$. PR$_{Bob}$= Key_Gen(Pub_Param, PR$_{Bob\_master}$, ID$_{Bob}$). The messages encrypted by PU$_{Bob}$ can be decrypted using PR$_{Bob}$.

```
Algorithm:
Host Alice Key Generation
    (a) $ID_{Alice} = IP_{Alice} || Port_{Alice}$
    (b) $Sign_{Alice}[En(S_k , request (Alice, PKG, ID_{Alice}))]$
    (c) $[Pub\_Param, PR_{Alice\_master}]=Sign_{PKG}[response(PKG, Alice)]$
    (d) $PR_{Alice} = Key\_Gen(Pub\_Param, PR_{Alice\_master}, ID_{Alice})$
Host Bob Key Generation
    (a) $ID_{Bob} = IP_{Bob} || Port_{Bob}$
    (b) $Sign_{Bob}[En(S_k , request (Bob, PKG, ID_{Bob}))]$
    (c) $[Pub\_Param, PR_{Bob\_master}]=Sign_{PKG}[response(PKG,Bob)]$
    (d) $PR_{Bob} = Key\_Gen(Pub\_Param, PR_{Bob\_master}, ID_{Bob})$
```

4.1.2 MPTCP SESSION KEY EXCHANGE USING KEY PAIR (MPC_SKEXMTCP)

4.1.2.1 STEP 1. SYN [MP_CAPABLE]

a) Encryption of Alice's key K_{Alice}: Alice encrypts the session key K_{Alice} with Bob's public key ($PU_{Bob} = ID_{Bob}$) using IBE.

b) Key Transmission of Alice. Alice sends the encrypted key $EK_{Alice}= En(PU_{Bob}, K_{Alice})$ to Bob with MP_CAPABLE.

4.1.2.2 STEP 2. SYN+ACK [MP_CAPABLE]

a) Encryption of Bob's key K_{Bob}: Bob encrypts the session key K_{Bob} with the public key of Alice PU_{Alice} using IBE.

b) Key transmission of Bob. Bob sends the encrypted key $EK_{Bob} = En (PU_{Alice}, K_{Bob})$ to Alice with MP_CAPABLE.

4.1.2.3 STEP 3. ACK [MP_CAPABLE]

a) Key Echoing: Alice sends the EK_{Alice} and EK_{Bob} again to complete the connection establishment.

```
Algorithm:
MPTCP 3-Way Handshake packet sequence
    (a) SYN (MP_CAPABLE) [$EK_{Alice}$, Flags] where $EK_{Alice} = En (PU_{Bob}, K_{Alice})$
    (b) SYN + ACK (MP_CAPABLE) [$EK_{Bob}$, flags] where $EK_{Bob} = En (PU_{Alice}, K_{Bob})$
    (c) ACK(MP_CAPABLE) [$EK_{Alice}$, $EK_{Bob}$ , Flags]
```

Figure 4.2 shows the packet sequence and the parameters passed with each packet of the initial 3-way handshake according to the proposed scheme. All over packet exchanges is shown in Figure 4.3.

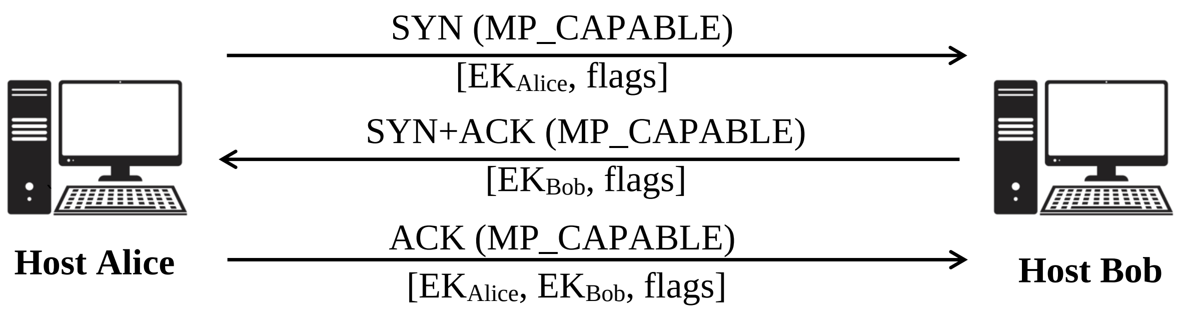

Figure 4.2 3-way handshake with the proposed scheme

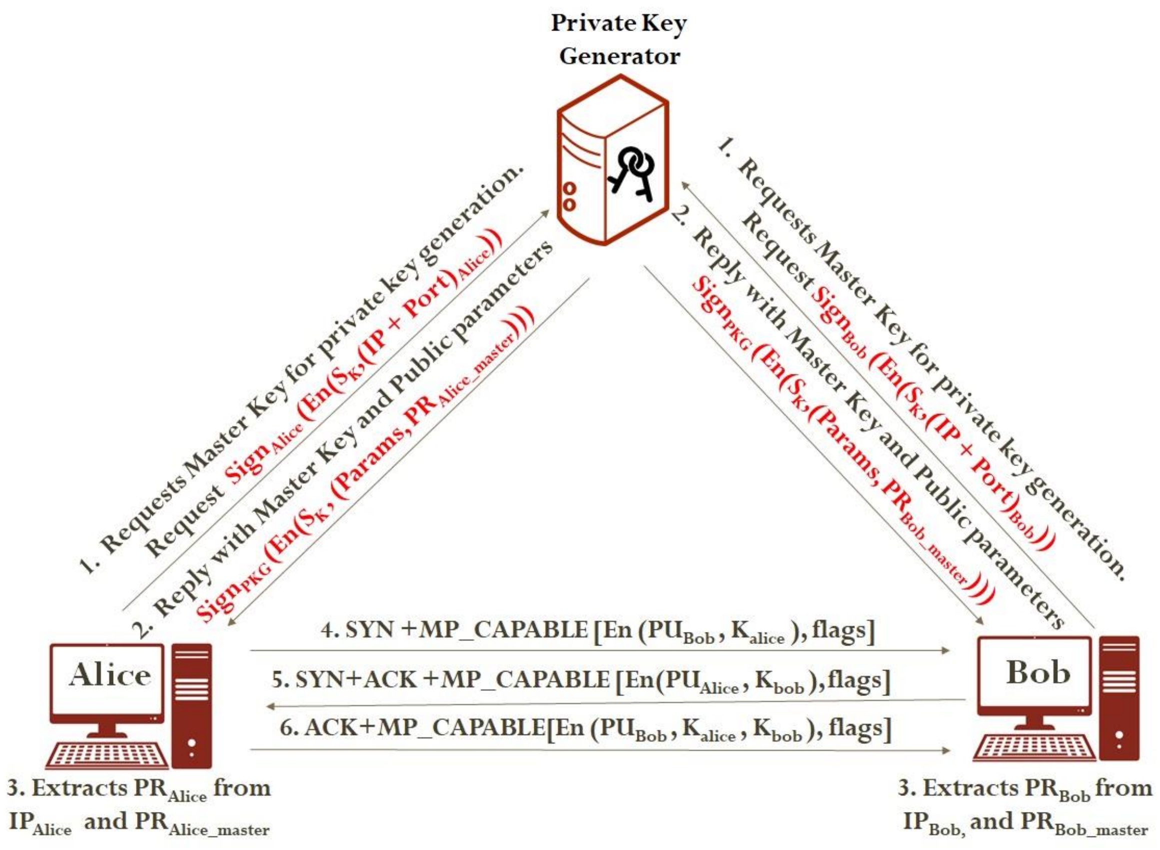
Private Key Generator
1. Requests Master Key for private key generation. Request $Sign_{Alice}(En(S_{K},(IP + Port)_{Alice}))$
2. Reply with Master Key and Public parameters $Sign_{PKG}(En(S_{K},(Params, PR_{Alice\_master})))$
1. Requests Master Key for private key generation. Request $Sign_{Bob}(En(S_{K},(IP + Port)_{Bob}))$
2. Reply with Master Key and Public parameters $Sign_{PKG}(En(S_{K},(Params, PR_{Bob\_master})))$
Alice
Bob
4. SYN +MP_CAPABLE [En (PU_{Bob}, K_{alice}), flags]
5. SYN+ACK +MP_CAPABLE [En(PU_{Alice}, K_{bob}),flags]
6. ACK+MP_CAPABLE[En (PU_{Bob}, K_{alice}, K_{bob}), flags]
3. Extracts PR_{Alice} from IP_{Alice} and PR_{Alice_master}
3. Extracts PR_{Bob} from IP_{Bob}, and PR_{Bob_master}

Figure 4.3 Secure key exchange during a 3-way handshake using IBE

Module 1 covers key generation with IBE, and module 2 covers the MPTCP 3-way handshake process; together, they provide the situation depicted in Figure 4.3. Bob must verify his identity to PKG when he asks for the private key to decrypt Alice's encrypted message. Bob uses his IP address and port number to verify his identity. PKG's session key generated via ECC and HMAC will be used to encrypt, and Bob's private key will be used to digitally sign the request for the private key (PRBob). Obtaining Bob's private key, required to forge the request packet, is extremely challenging. The attacker needs to perform at least 2n tries to crack the security of a system where the private key is n bits in length.

4.2 SECURE COMMUNICATION BETWEEN PKG AND COMMUNICATING HOSTS

The ECC is used to secure the communication between hosts and PKG for the exchange of public parameters and private keys for IBE. The ECC is based on the elliptic curve $E_p(a,b)$ over finite field $GF(2^m)$, which satisfies the below cubic equation [34] [36].

$$y^2 = x^3 + ax + b,$$

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 the 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.

The ECC can be defined using the DLP and the DH. ECDHP is a key agreement protocol that works upon Diffie Hellman's concept over elliptic curves. ECDLP is an extended version of DLP over a finite field. ECDLP and ECDHP are used to secure the connection between the communicating hosts and PKG.

ECDLP can be defined by considering equation $Q = kP$, the discrete logarithm of Q to the base P, for the given points P and Q in group $E_p\ (a,b)$ And $k < p$ It will be exceptionally challenging for the adversary to figure out 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 is the integer of the form 2^m. In addition, select the coefficients a and b for the elliptic curve $E_q(a,b)$. Now, select point $G = (x_1, y_1)$ On the elliptic curve of the order of a large number n. Here, $E_q(a,b)$ and G are the global parameters, which will be used for the key generation and will be known to all the participants. The key exchange process for Alice and PKG and Bob and PKG is as follows:

a) 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.

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

b) By using the same equation, PKG can select the private key n_{PKG} And generate the public key P_{PKG}.

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

d) The session key for communication between Alice and PKG is as below:

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

The same process can generate the secret key for secure communication between Bob and PKG, as shown in Figure 4.4.

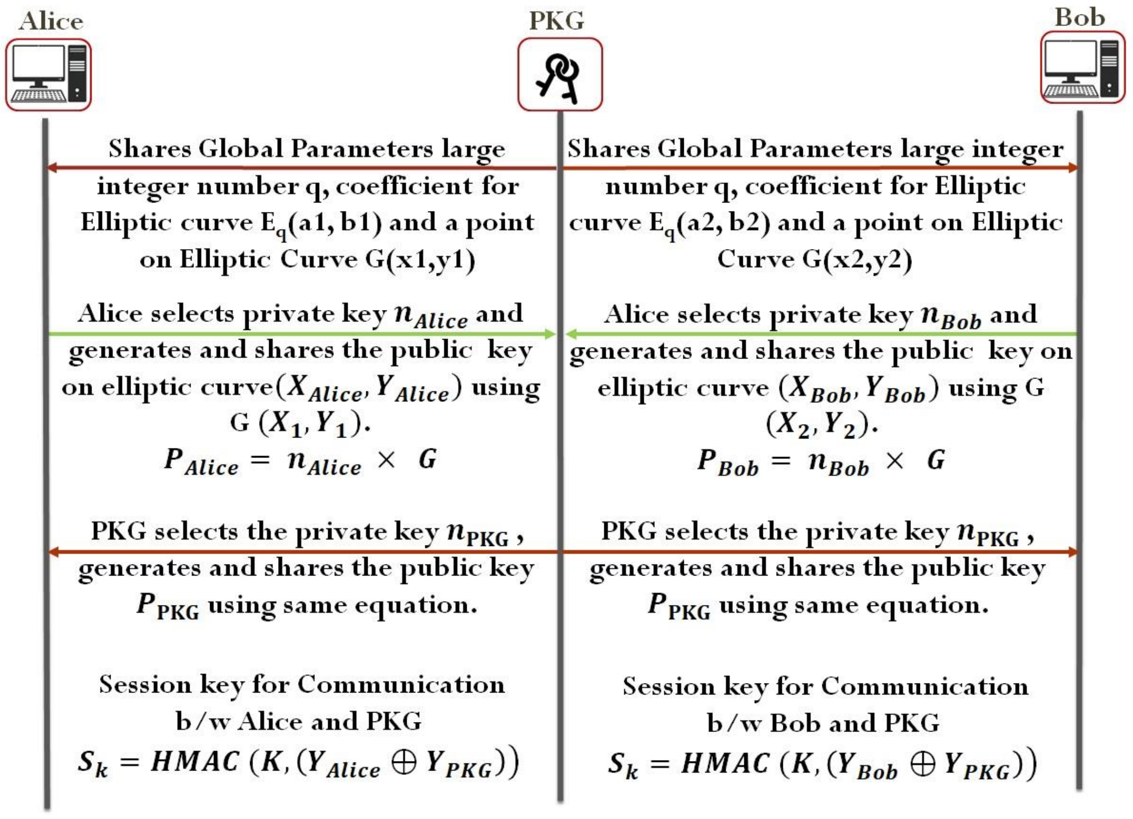
Alice
PKG
Bob
Shares Global Parameters large integer number q, coefficient for Elliptic curve E_q(a1, b1) and a point on Elliptic Curve G(x1,y1)
Shares Global Parameters large integer number q, coefficient for Elliptic curve E_q(a2, b2) and a point on Elliptic Curve G(x2,y2)
Alice selects private key n_{Alice} and generates and shares the public key on elliptic curve(X_{Alice}, Y_{Alice}) using G (X_1, Y_1).
$P_{Alice} = n_{Alice} \times G$
Alice selects private key n_{Bob} and generates and shares the public key on elliptic curve (X_{Bob}, Y_{Bob}) using G (X_2, Y_2).
$P_{Bob} = n_{Bob} \times G$
PKG selects the private key n_{PKG}, generates and shares the public key P_{PKG} using same equation.
PKG selects the private key n_{PKG}, generates and shares the public key P_{PKG} using same equation.
Session key for Communication b/w Alice and PKG
$S_k = HMAC\ (K, (Y_{Alice} \oplus Y_{PKG}))$
Session key for Communication b/w Bob and PKG
$S_k = HMAC\ (K, (Y_{Bob} \oplus Y_{PKG}))$

Figure 4.4 Key exchange scenario between Alice–PKG and Bob–PKG