MPCTP Residual Threat Analysis

draft-bagnulo-mptcp-attacks-00

M. Bagnulo, C. Paasch, F. Gont, O. Bonaventure, C. Raiciu

MPTCP WG meeting - IETF87

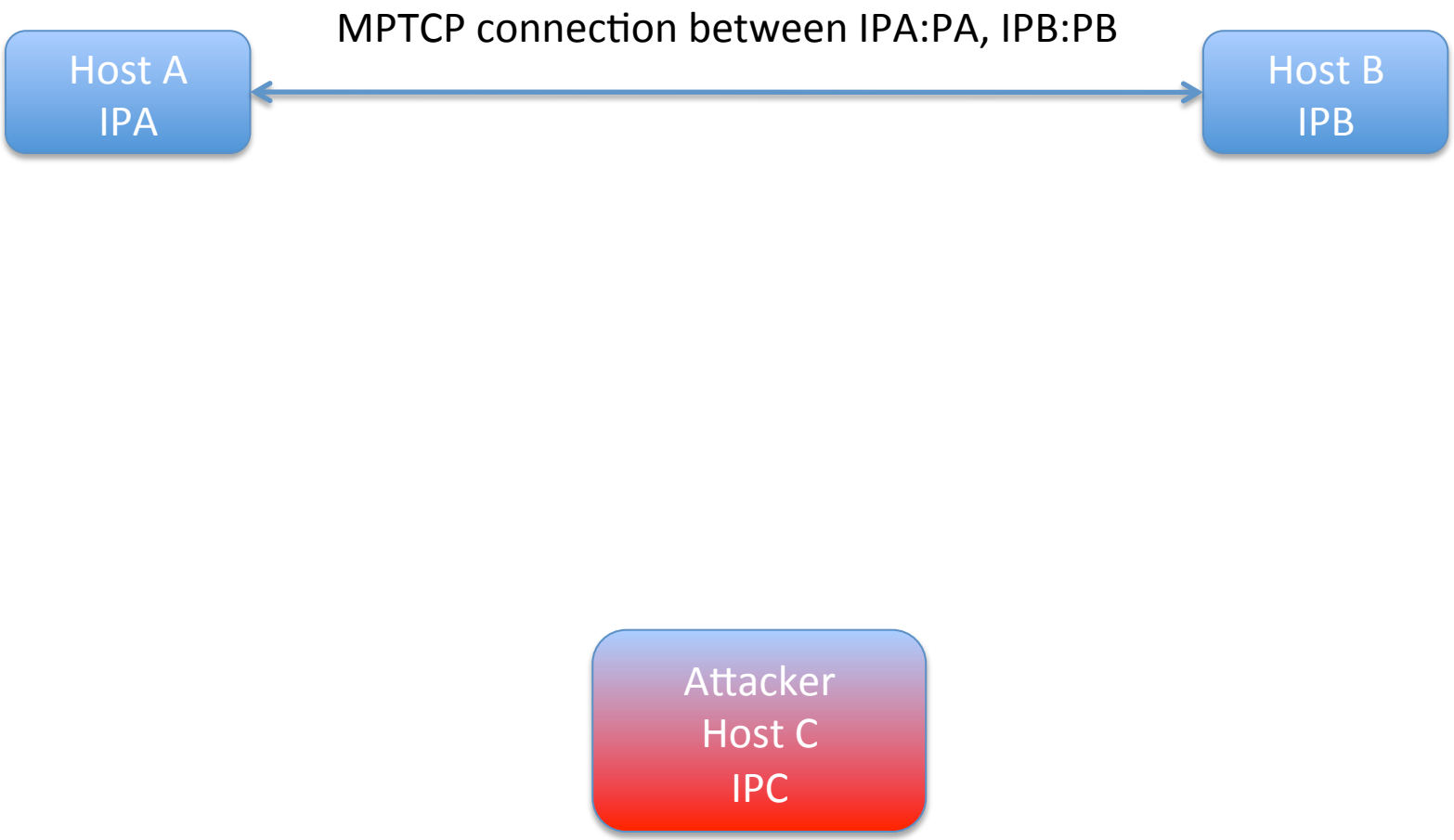
Background

- RFC6181 – “Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses” provided general threat analysis prior RFC6824
- This effort is after RFC6824 is done, figure out what residual threats exist.
- Goal: (for both of them) Make MPTCP no worse than current unipath TCP
  - NOT a goal to improve TCP security

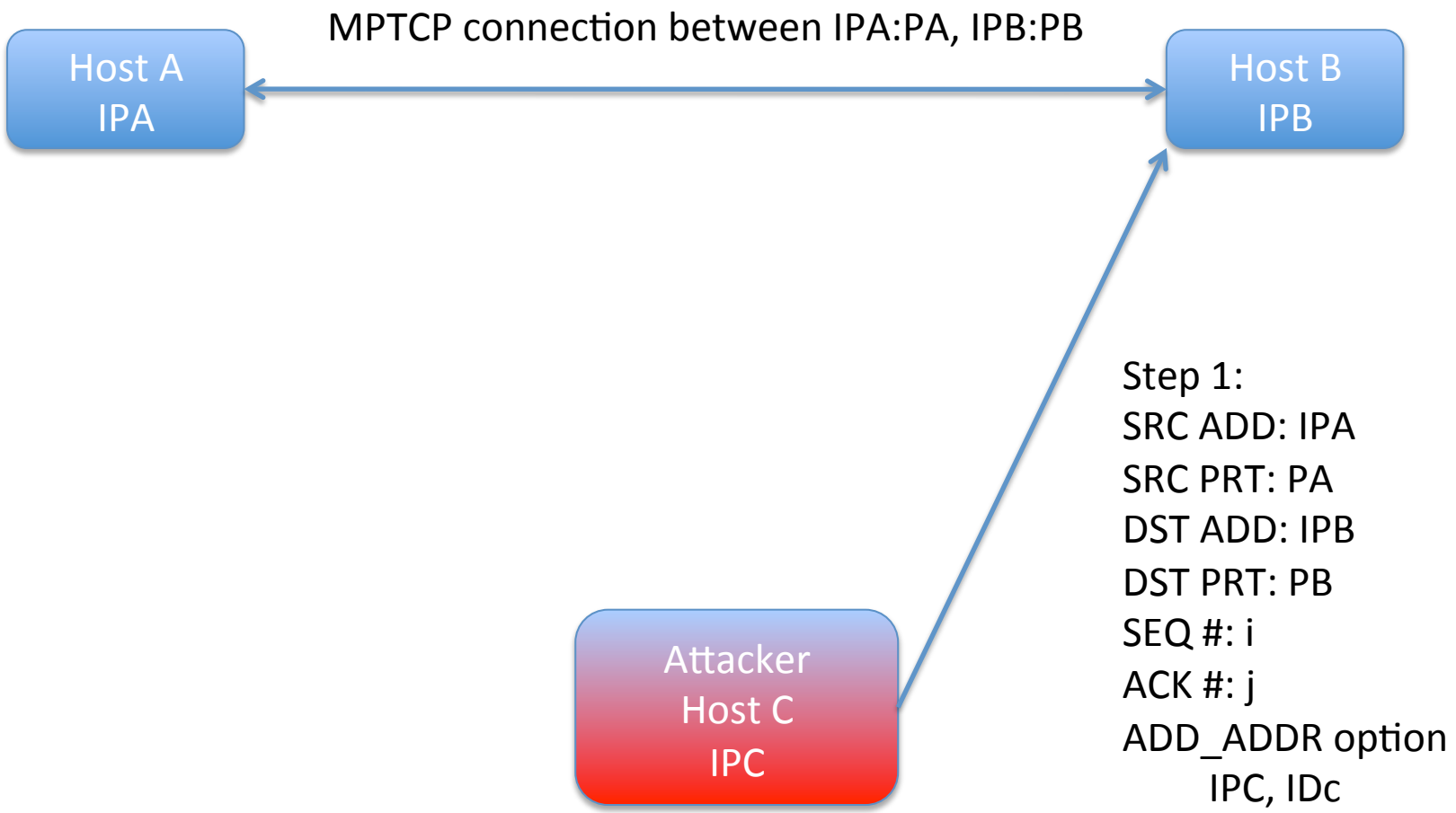
Attacks identified so far…

- ADD_ADDR attack
- SYN+MP_JOIN DoS attack
- SYN flooding Amplification
- Eavesdropper in the initial handshake (known)

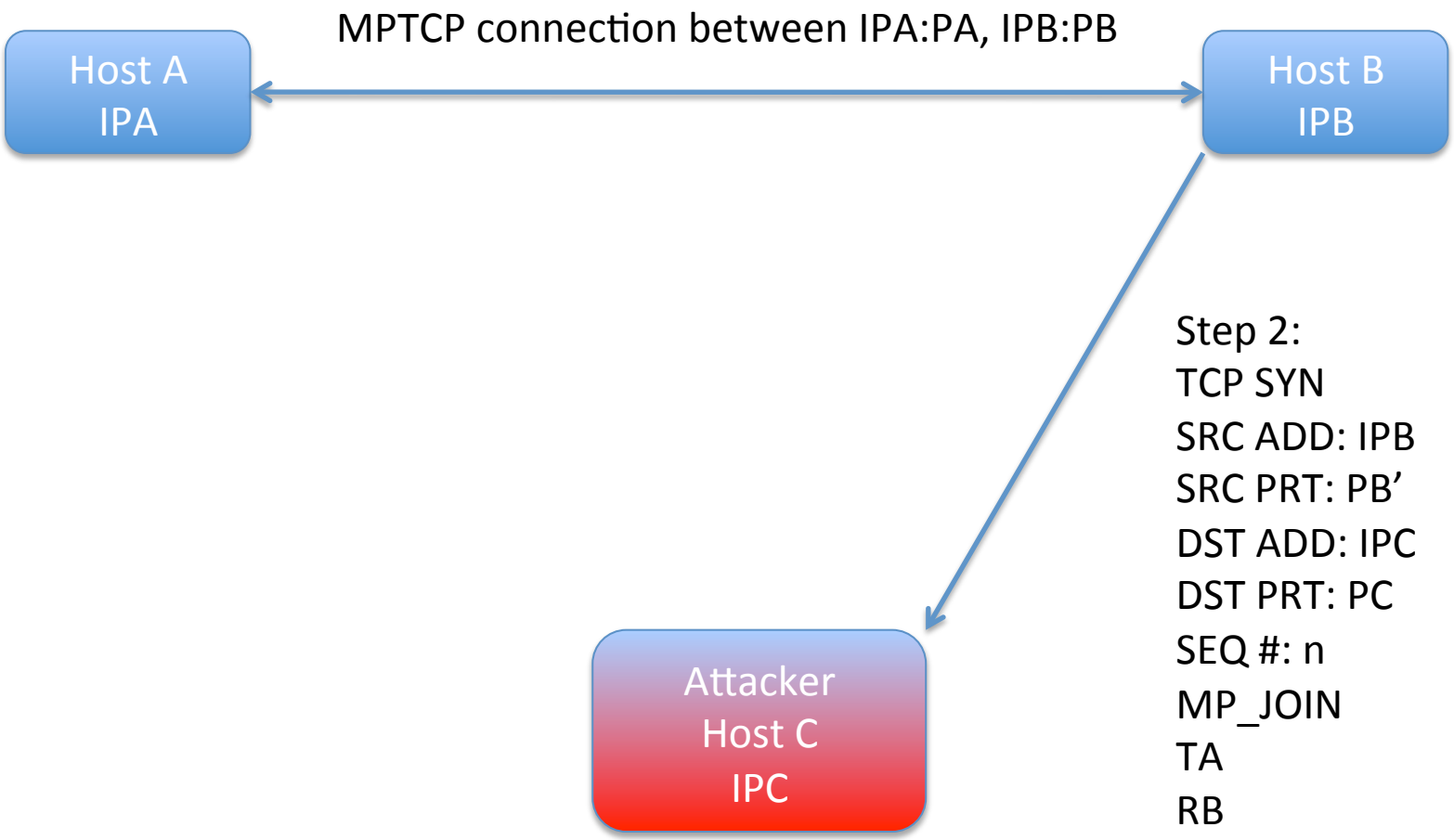
ADD_ADDR attack



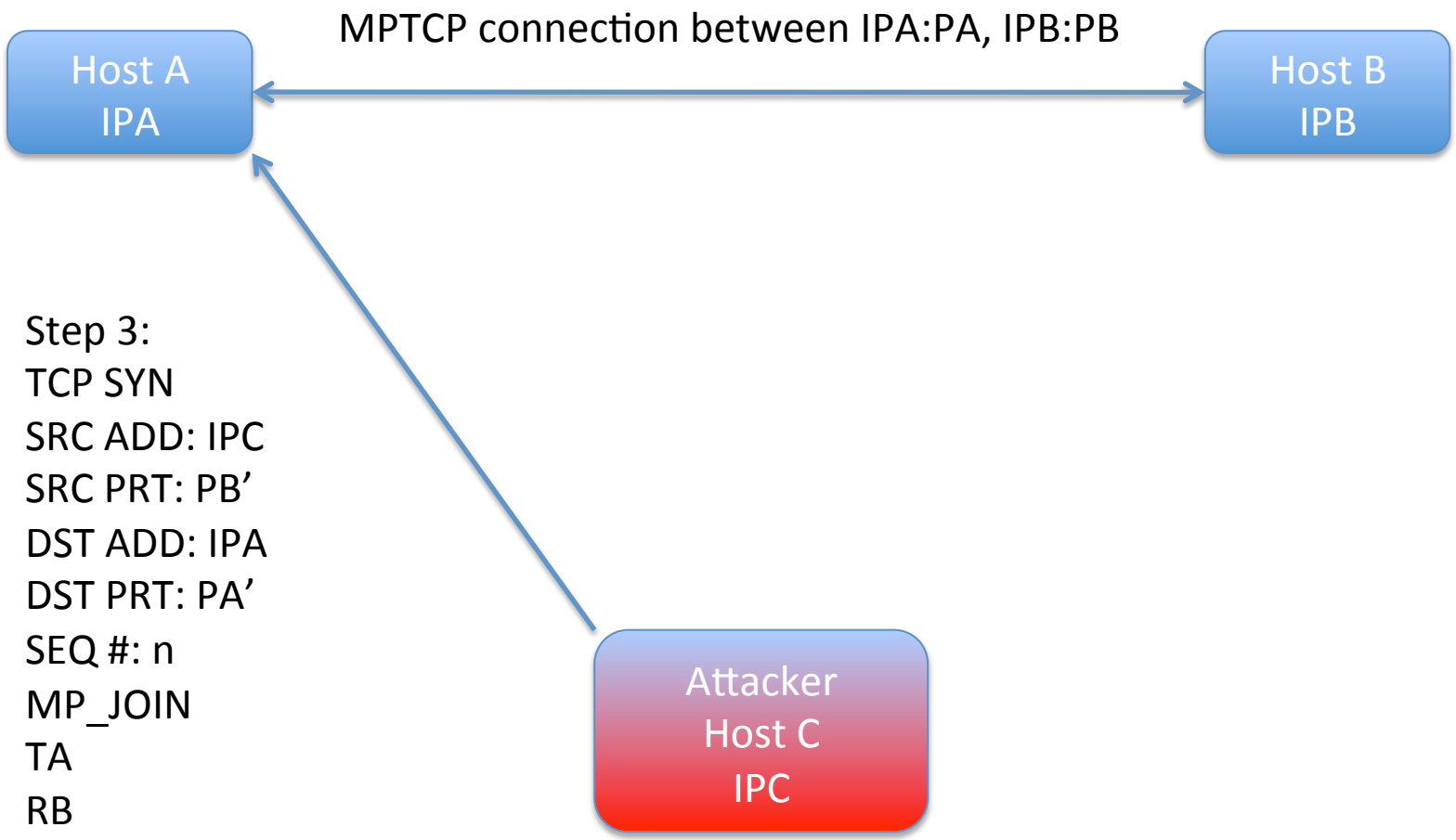
ADD_ADDR attack



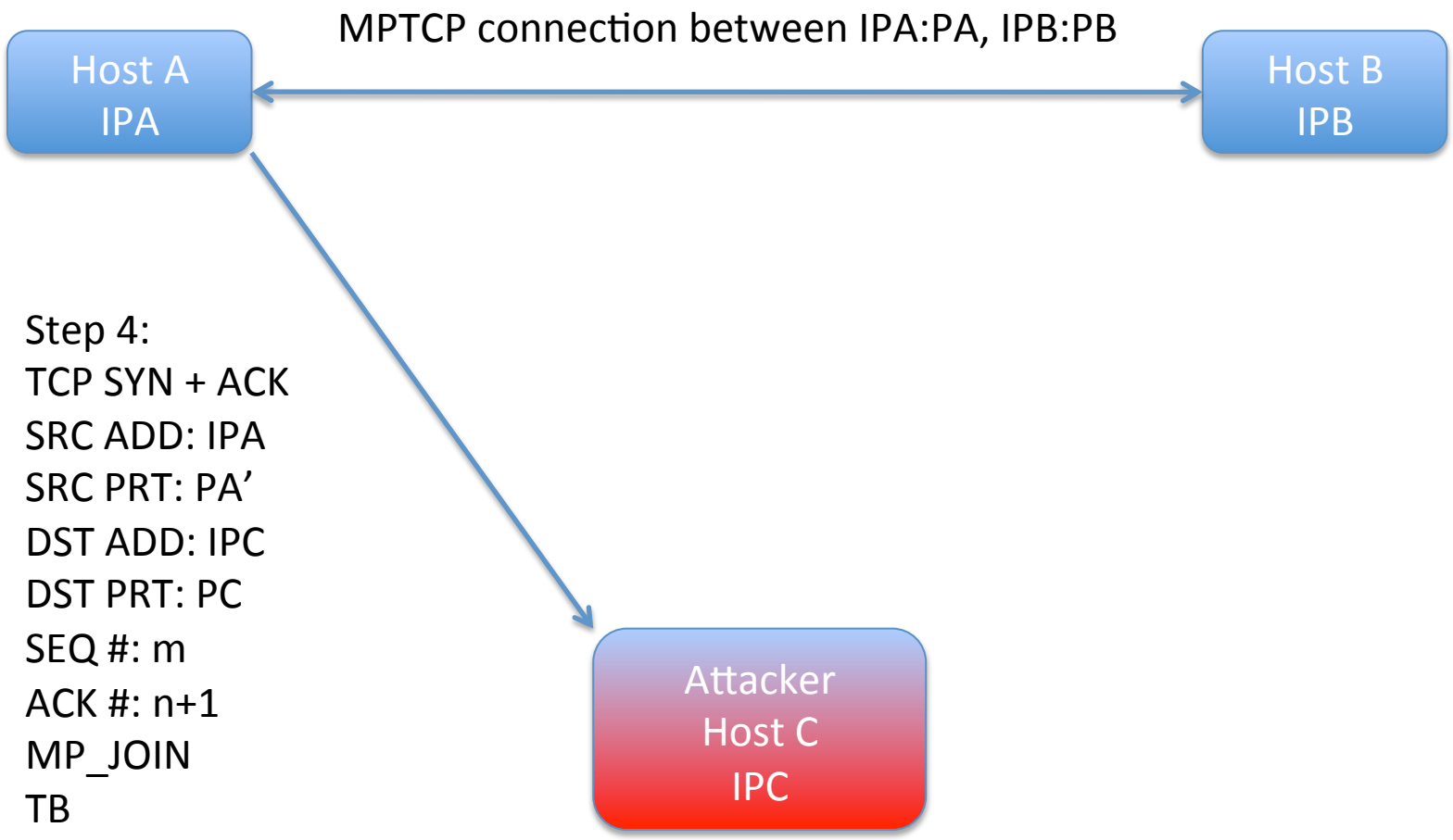
ADD_ADDR attack



ADD_ADDR attack

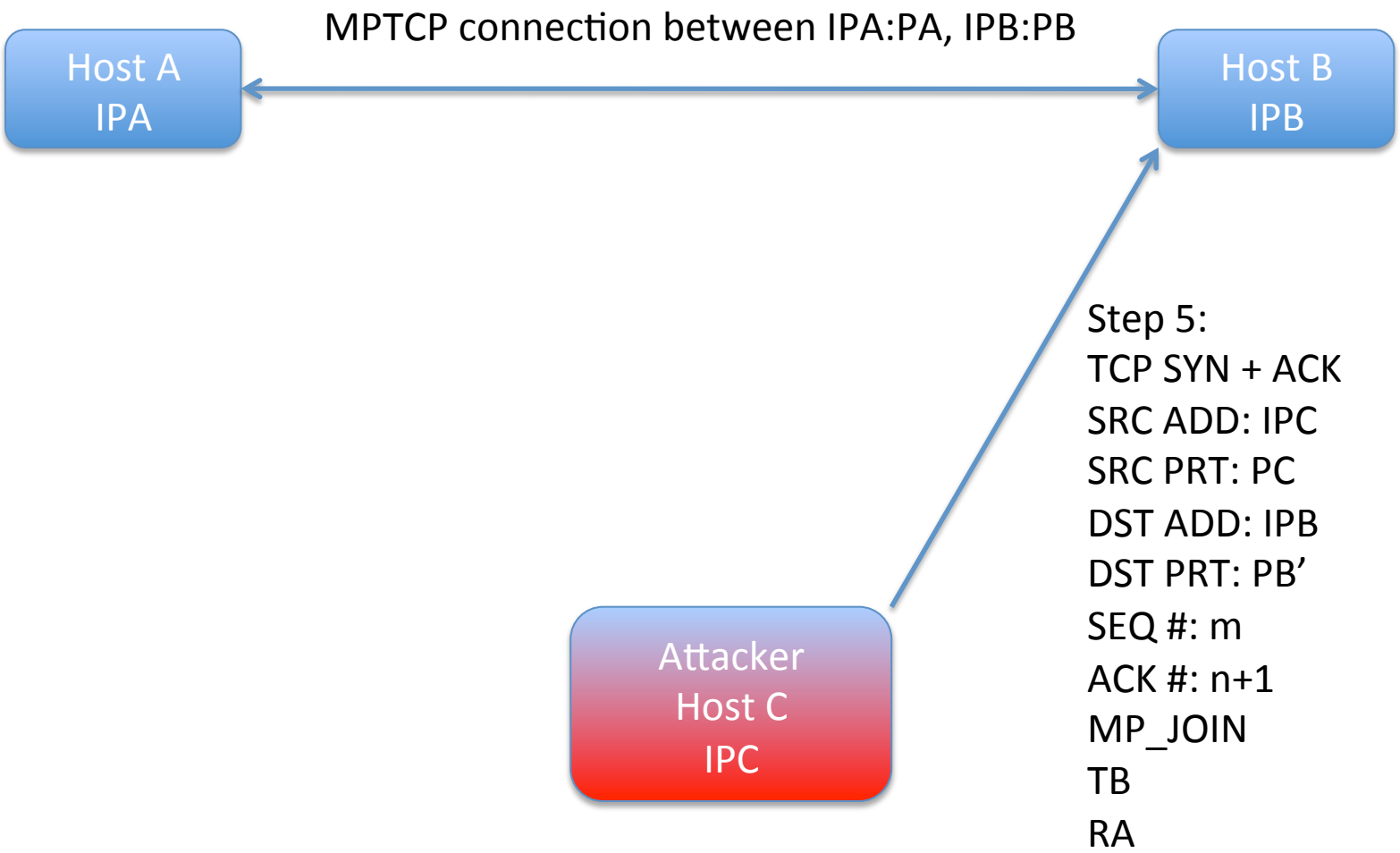


ADD_ADDR attack

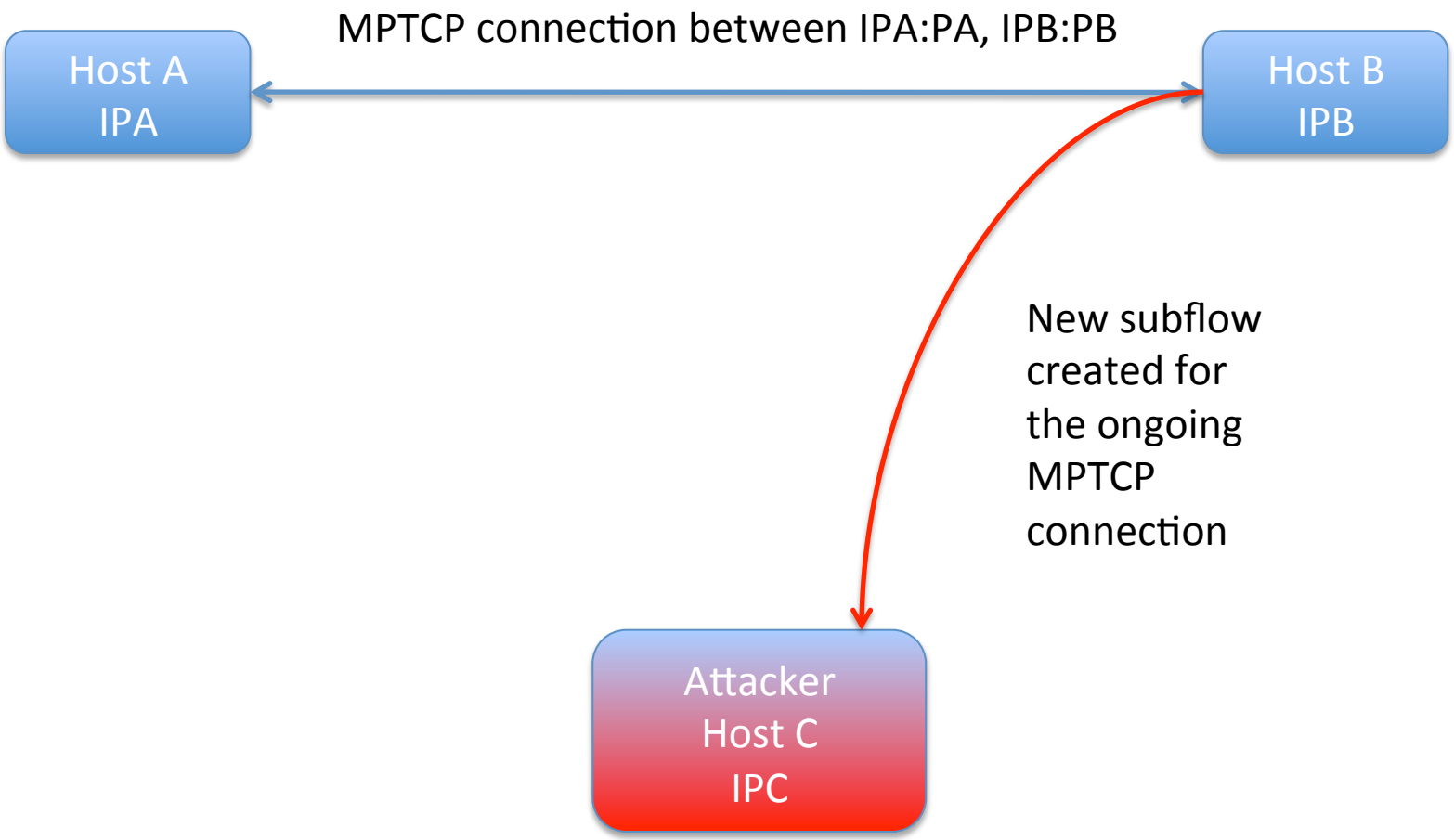


Step 4:
TCP SYN + ACK
SRC ADD: IPA
SRC PRT: PA'
DST ADD: IPC
DST PRT: PC
SEQ #: m
ACK #: n+1
MP_JOIN
TB
RA

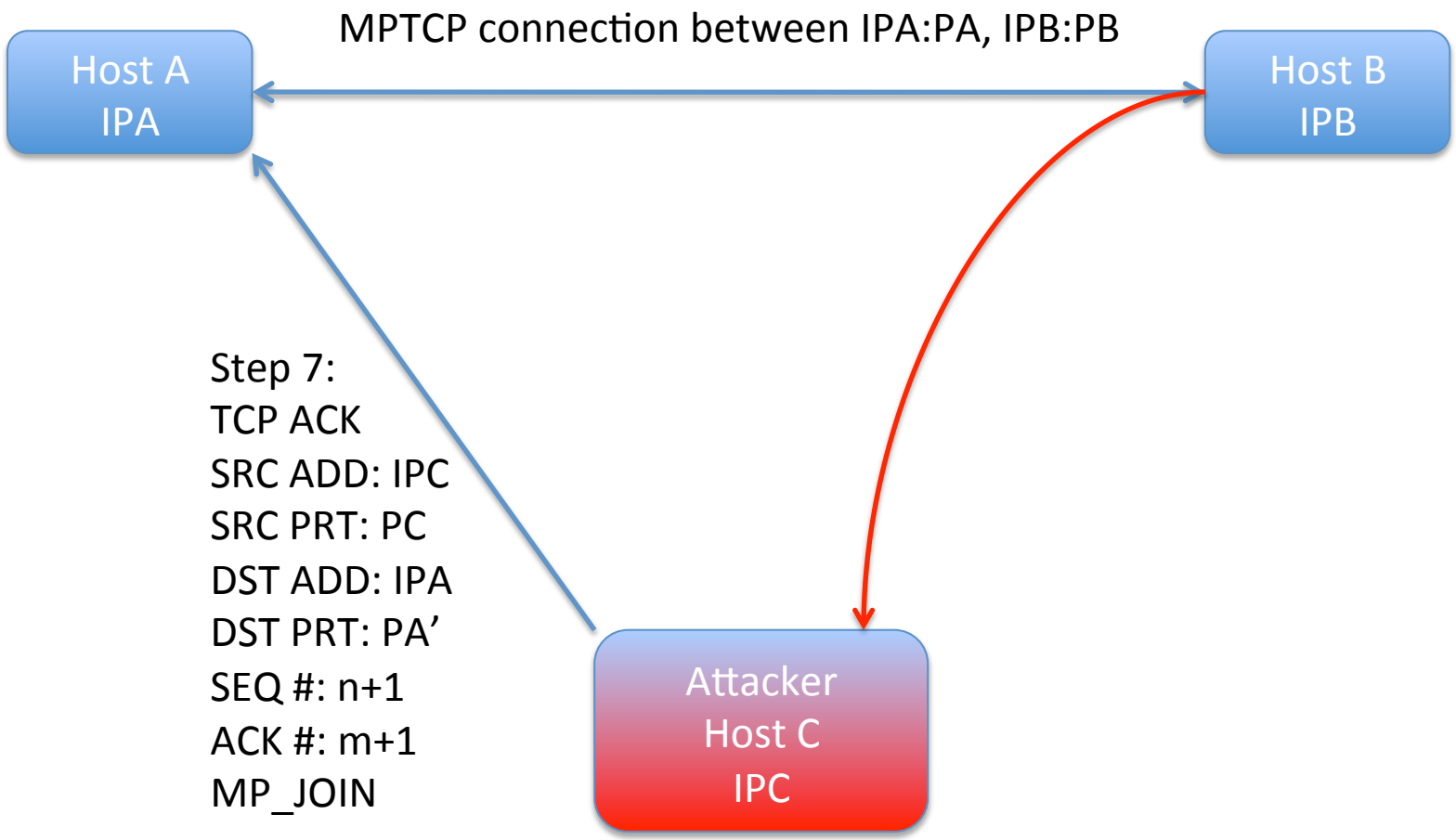
ADD_ADDR attack



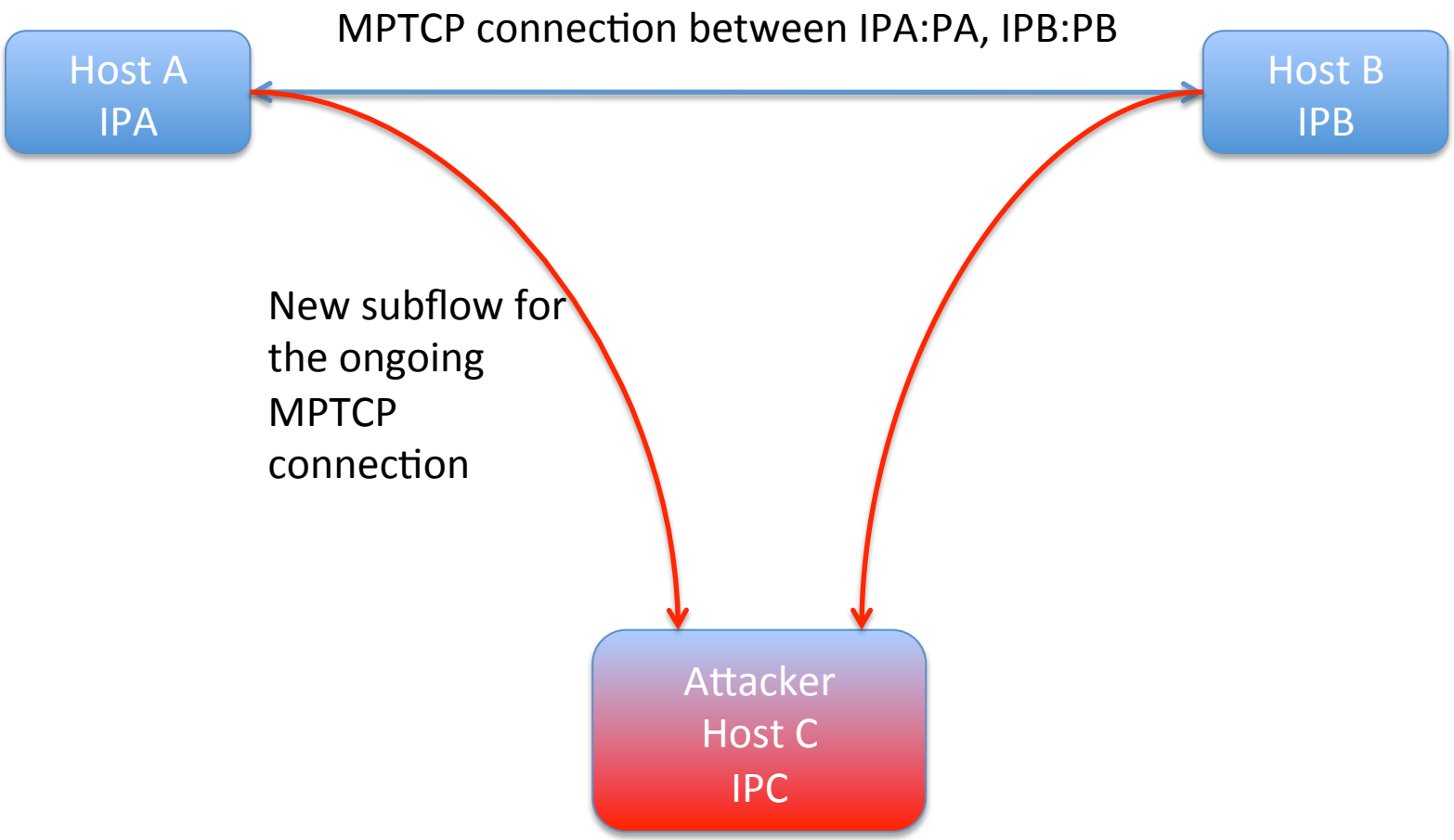
ADD_ADDR attack



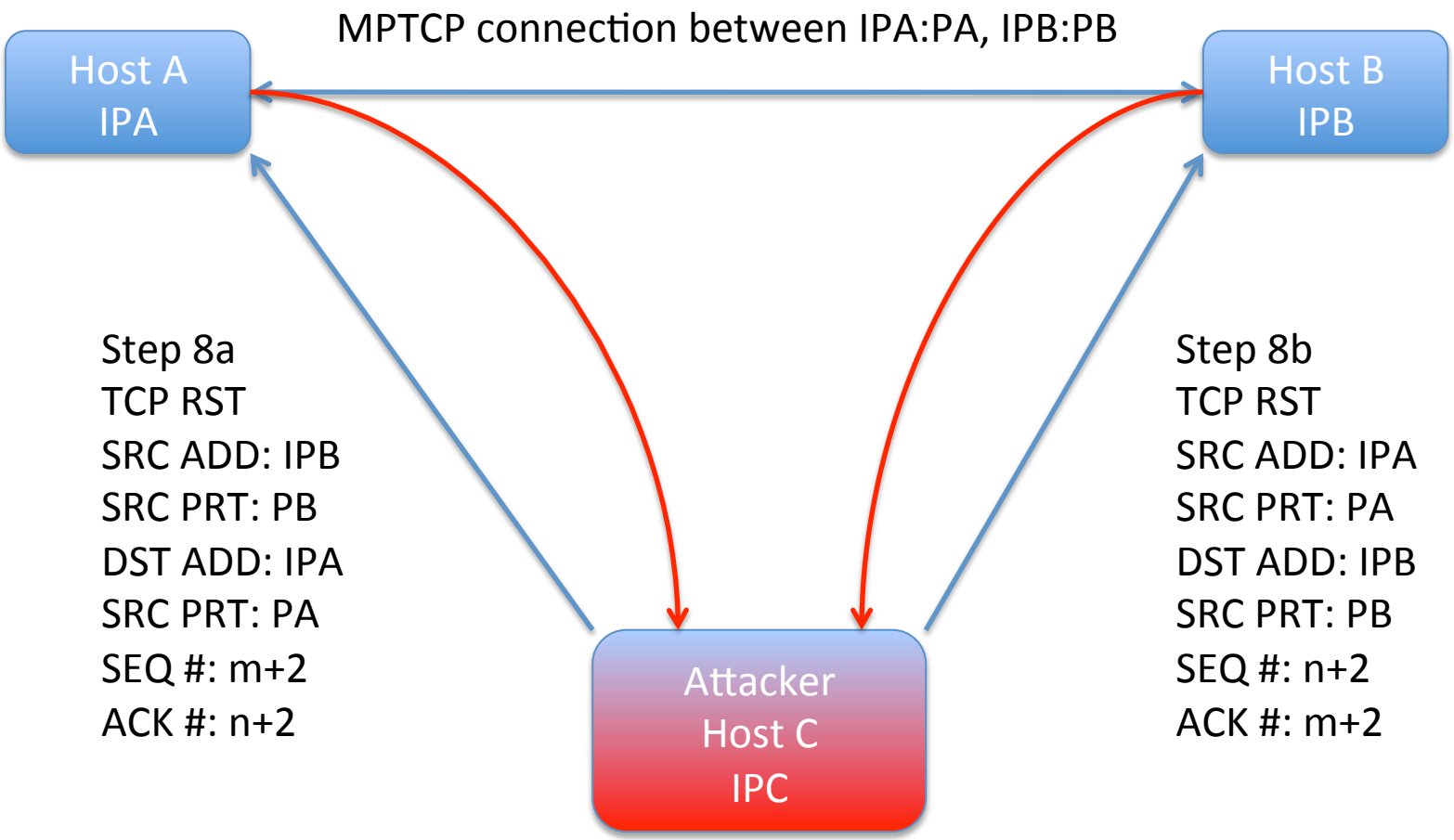
ADD_ADDR attack



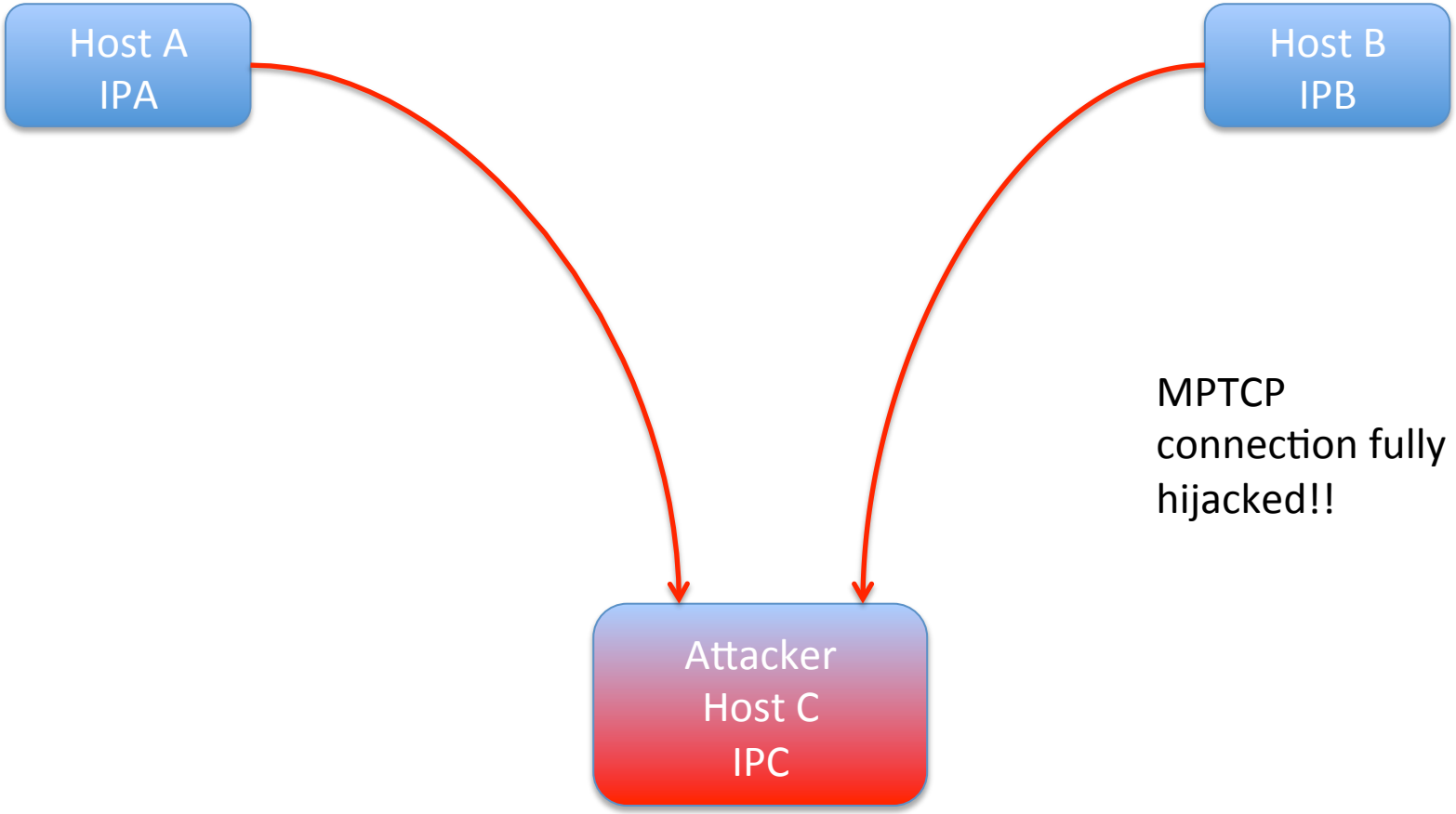
ADD_ADDR attack



ADD_ADDR attack



ADD_ADDR attack



MPTCP connection fully hijacked!!

ADD_ADDR attack

- Information needed
  - IPA:PA, IPB:PB
    - The attacker needs to know that a MPCTP connection exists
      - If MPTCP use is pervasive, then it is likely it will be easy to guess, e.g. with popular services.
      - The server port may be easier to guess than the client port. For the client port, it may be necesary to explore all the ephemeral ports?
  - IDc
    - Avoid collision, should be very easy
  - SEQ #: should be within the RCV window of Host B
  - ACK #: should be within the SENDER WND of Host B.

Possible solutions for the ADD_ADDR attack

- Solution 1: include the 32-bit token in the ADD_ADDR option
  - Weak solution (but may be enough?): attacker needs to know the token
- Solution 2: include an HMAC of the key of the RCVR in the ADD_ADDR
  - Requires the attacker to figure out the key. (need to make a comparative analysis of how harder is for the attacker to find out the key compared to the token)

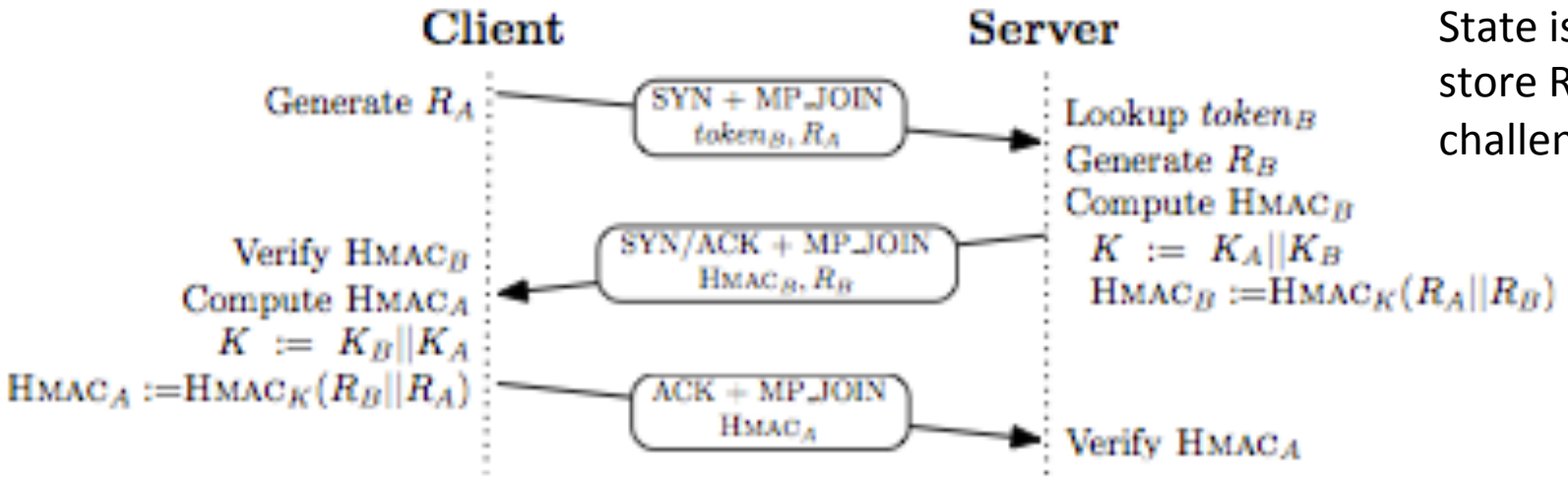
Possible solutions for the ADD_ADDR attack (cont.)

- Solution 3: Include an HMAC of the added address in the ADD_ADDR
  - Implies that the host sending the ADD_ADDR cannot be behind a NAT
- Solution 4: Include an HMAC of the DST address in the MP_JOIN
  - Again, need to assess the impact of NATs

SYN+MP_JOIN DoS attack

- SYN flooding attack is DoS attack sending multiple SYNs and requiring the server to create state on the SYN
  - Solved by deferring state creating till the ACK
- IN MPTCP, state is created after the reception of the SYN+MP_JOIN, enabling a DoS attack.
- The effect of the attack is that the legit host cannot create new subflows
-

SYN+MP_JOIN DoS attack



State is created to store R_A and the challenge R_B

The attacker needs to know the token B in order to perform the attack

32-bit random number

One option is for an attacker to create a MPTCP connection

Possible solution for the SYN +MP_JOIN DoS attack

- Solutions:
  - the subflow creation is as expensive for the attacker than for the victim
    - The state is created after the ACK. Question: how does the sever verifies the challenge? The challenge could be created as a hash of known thinks (e.g. The addresses and ports int he SYN+MP_JOIN msg
  - the state is created after the initiator has proven to have the key
    - Similar to the previous one

SYN flooding Amplification

- The attacker to create a MPTCP connection using a legit address and then send multiple SYN+MP_JOIN with other (non valid) addresses.
  - This is a DoS attack that would consume more resources in the victim than the ones required by the attacker.
- Possible solution: limit the number of half open subflows.

Other attacks

- Eavesdropper in the initial handshake.
  - Can sniff the key and then take over the MPTCP connection.
  - Solutions:
    - Hash chains
    - SSL/DH
    - TCPCrypt
    - CGAs
    - DNSSEC