Secure MPTCP

draft-bagnulo-mptcp-secure-00

M. Bagnulo

IETF89

Motivation

- MPTCP (RFC6824 and RFC6824bis) is more vulnerable than TCP
  - Passive eavesdropper during initial handshake can hijack the MPTCP connection AFTER leaving the on-path location
  - Active attacker can hijack the connection by subverting a JOIN msg and then can redirect the connection to an IP addres of its will.
  - In regular TCP, the attacker must be on path all the time during the atttack

Motivation (2)

- draft-ietf-mptcp-attacks concludes that to address these attacks, the data stream should be protected (rrather than the MPTCP control msgs)
  - The reason for this is NAT compatibility
- tcpcrypt is one approach to secure the payload, it is natural to explore the MPTCP/ tcpcrypt integration (hereafter called SMPTCP)

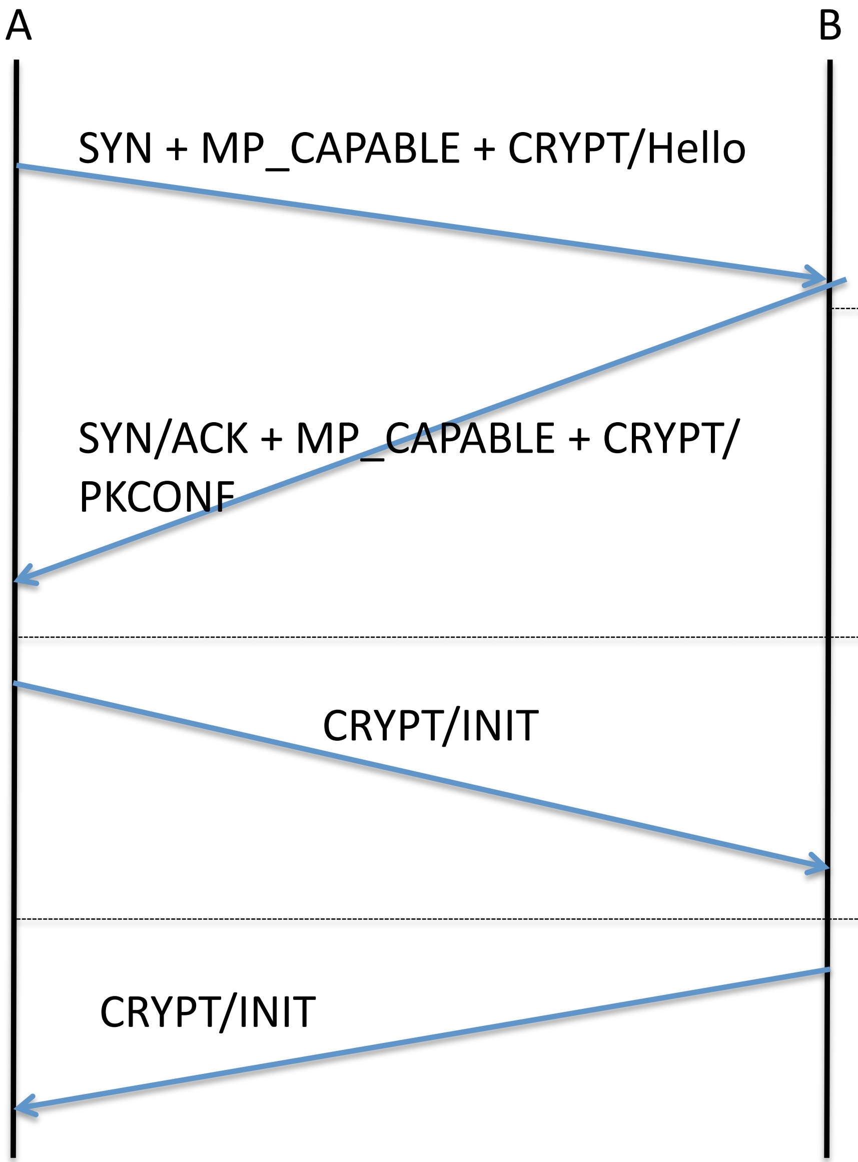
Motivation (3)

- In addition MPTCP shares the vulnerabilities of TCP
  - Passive eavesdropping
  - Insertion of content in data stream
  - DoS attacks (RST spoofing, ack spoofing, dropping packets)
- tcpcrypt naturally address at least the first two attacks, so better than TCP security will result

Limitations and issues

- tcpcrypt does not protect against MitM attacks, so SMPTCP will be vulnerable to MitM
  - the attacker must be active and on path attacker to eavesdrop
  - It needs to be active to hijack and needs to block all subflows between legitimate endpoints

SMPTCP initial handshake



MP_CAPABLE carries A's key (ka) for backward compatibility
15 bytes in total
MSS, SACK, WND scale and tiemstamps option fit

MP_CAPABLE carries B's Key (kb) for uniqueness (discuss)
PKCONF carries pub-cipher-list (2 alg are 10 B) making total of 22 B in total
Not all the above options would fit.

Crypto material in payload
3 B of options
MP_CAPABLE not needed as tcpcrypt keys are used

Crypto material in payload
3 B of options

Material resulting after handshake

MPTCP

- One session key per each side (Key A and Key B), to secure HMACs
- One token per each side, to identify the connection
- One IDSN per each side

tcpcrypt

- Encryption keys, one per each side (kec, kes)
- Authentication keys, one per each side (kas, kac)
- A session ID (SID) of 64 bits, statitically unique

Generating MPTCP values out of tcpcrypt values:

•Key A = kac

•Key B = kas

•Token A = 32msb(hash(ka)) - ka exchanged in MP_CAPABLE

•Token B = 32msb(hash(kb)) - kb exchanged in MP_CAPABLE

•IDSN A = 64lsb(hash(kac+SID))

•IDSN B = 64lsb(hash(kas+SID)

Adding subflows / Adding addresses

- All subflows protected with the same keys (kac, kas)
- Instead of using the original keys in MPTCP the news keys are used for the HMAC protection of JOIN and ADD-ADDR messages.

Exchanging data

- tycpcrypt adds a MAC option in each TCP segment
  - 160bit long MAC, 22 bytes of option
- MPTCP adds the DSS option
  - The DSS option doesnt have to be carried in every packet, but it can be
    - What is a reasonable stratehy?

DSS option

- Maximum length: 28B
  - Plus 22B from MAC option > 40B
- Ways to deal with this:
  - DSS option includes both DSN to seq# map and data ack
    - DSS option only with Data ack, 12B. 12B + 22B <40B
    - DSS option only with DSN to seq# map, 20B
      - Still a problem, 20B + 22 B> 40B
      - We could remove the 2B checksum when tcpcrypt is used
        - » After all, there is a 22B MAC!
      - We could use 4B data seq numbers
        - » DSS option with Data ack and 4B DSNs: 8B
        - » DSS option with DSN to seq # map with 4B DNSs: 12 B

Backward compatibility

Types of nodes

- MPTCP nodes
- tcpcrypt nodes
- SMPTCP nodes
- Legacy nodes
- MPTCP/tcpcrypt

Expected behaviour

	SMPTCP	MPTCP	tcpcrypt	legacy	MPTCP/ tcpcrypt
SMPTCP	SMPTCP	MPTCP	tcpcrypt	TCP	??
MPTCP		MPTCP	TCP	TCP	MPTCP
tcpcrypt			tcpcrypt	TCP	tcpcrypt
legacy				TCP	TCP
MPTCP/ tcpcrypt					??

Challenges

- tcpcrypt uses 22 B of options in every segment. When used with MPTCP, we use all option sapce for some segments
  - No room left for SACK, and others
  - Would it be possible to use less option space?