Application Layer Authentication for MPTCP

Christoph Paasch <cpaasch@apple.com>
Alan Ford <alan.ford@gmail.com>

draft-paasch-mptcp-application-authentication
draft-paasch-mptcp-tls-authentication

RFC 6824 handshake

- Key is sent in plaintext
  - Easy for attacker to hijack a session
- Token generation
  - Hash-collisions introduce computational overhead
  - Load balancers would need to maintain state

Current Handshake

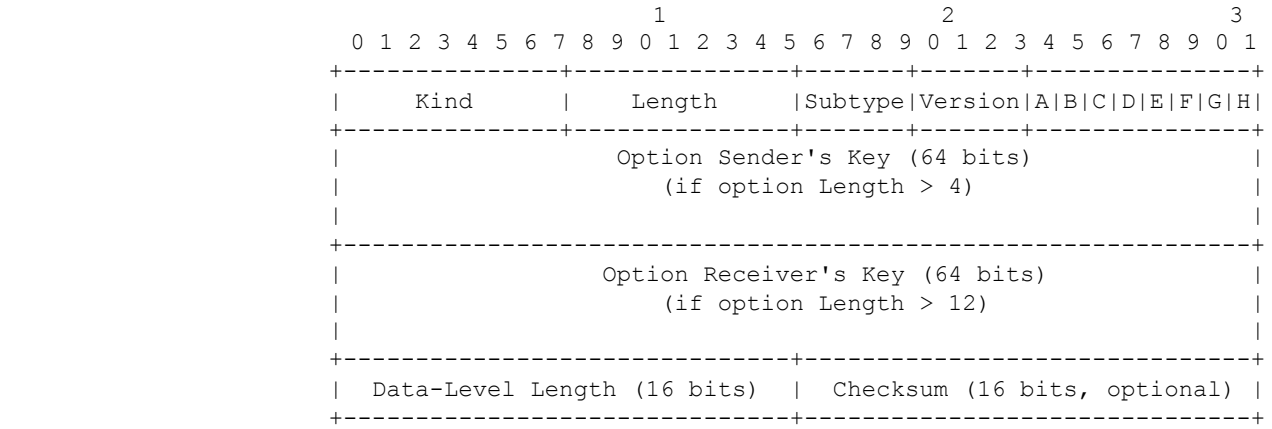


Figure 4: Multipath Capable (MP_CAPABLE) Option

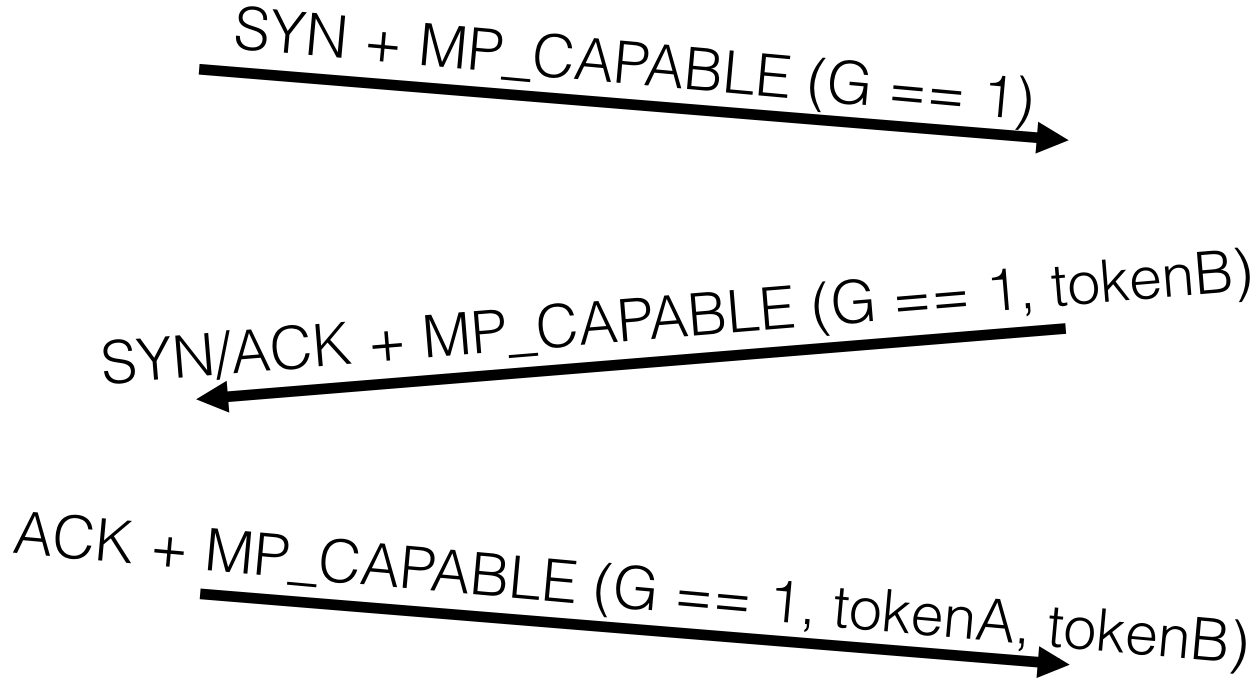
- SYN (A->B): only the first four octets (Length = 4).
- SYN/ACK (B->A): B's Key for this connection (Length = 12).
- ACK (no data) (A->B): A's Key followed by B's Key (Length = 20).
- ACK (with first data) (A->B): A's Key followed by B's Key followed by Data-Level Length, and optional Checksum (Length = 22 or 24).

Goals

- Make token explicit in the MP_CAPABLE handshake
  - Allows uniqueness of the token without trial-and-error approach
  - Enables token to carry information for load balancers
- Allow external keys to be fed into MPTCP
  - Prevents hijacking attacks on MPTCP

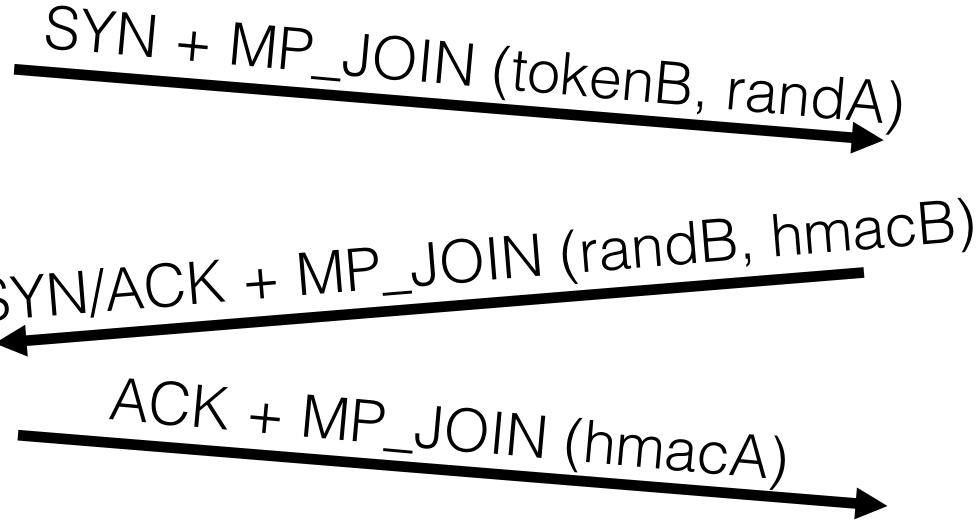
MP_CAPABLE handshake

- Use the G-bit to indicate key-derivation from the application
- Minimal change to 6824bis



MP_JOIN handshake

- Application provides keyA and keyB to the MPTCP-stack
- Same handshake as RFC 6824



hmacA = hmac(keyA + keyB, randA + randB)
hmacB = hmac(keyB + keyA, randB + randA)

Integration with TLS

- *draft-paasch-mptcp-application-authentication* defines the "G" bit and thus the exchange of tokens not keys in the MP_CAPABLE handshake
  - Proposed for inclusion in 6824bis

- *draft-paasch-mptcp-tls-authentication* shows how to use this with TLS – use of RFC5705 key exporters for exchanging the key
  - Application-layer decision. Separate from 6824bis.

Summary

- RFC 6824bis already changed the handshake to enable reliable stateless web servers
- Our minor modification enables:
  - better scalability
  - better security
  - easier deployment behind load balancers