Securing the Multipath TCP handshake with external keys

Christoph Paasch
Olivier Bonaventure

draft-paasch-mptcp-ssl-00 (expired)

Motivation

- RFC 6824 sends the keys in clear
  - Attacker who sees the initial handshake can hijack an MPTCP session

- TCPcrypt could help, but it is not always necessary (e.g., SSL/TLS)

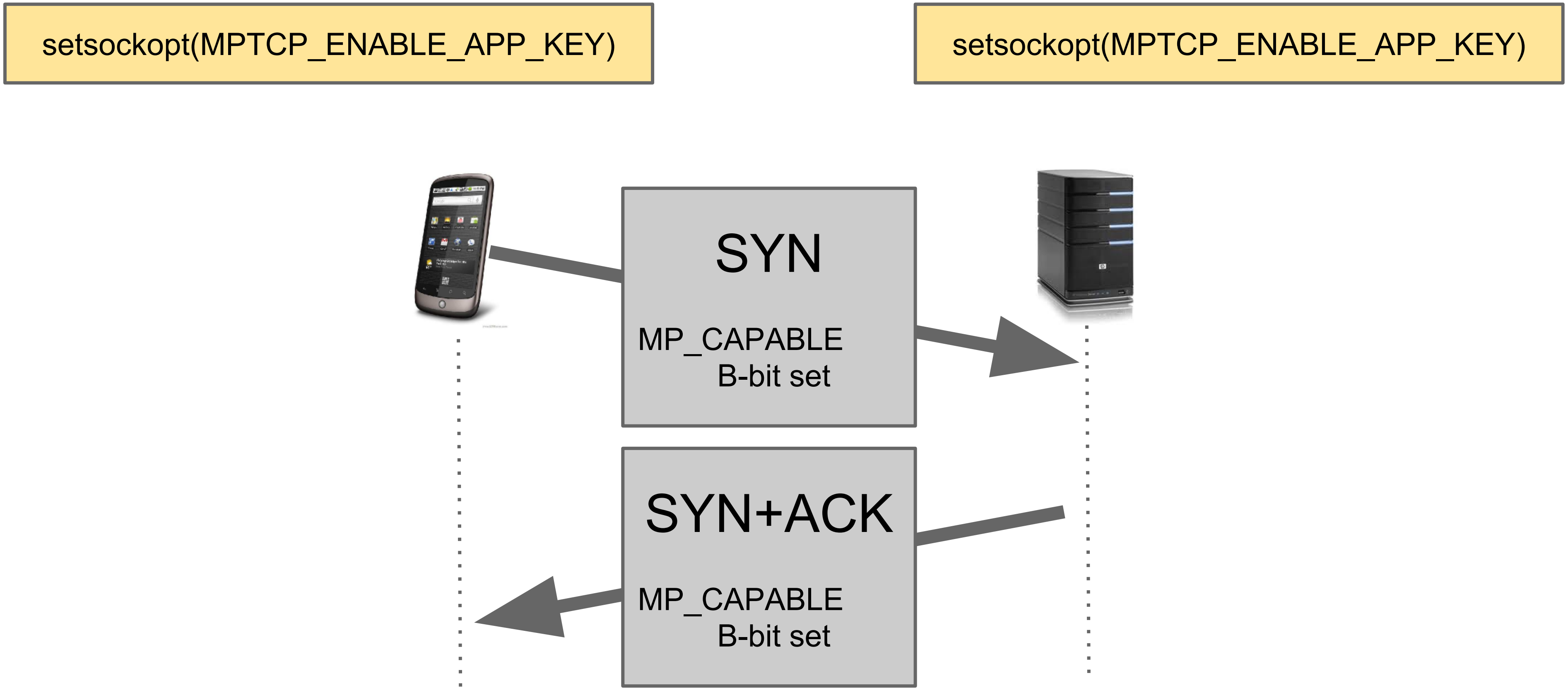
Securing MPTCP with external keys

- Application-level protocols already do negotiate a key (SSL/TLS)

  We should use these keys!

- Extend the socket-API to allow keys from the application

SSL initial handshake



setsockopt(MPTCP_ENABLE_APP_KEY)

setsockopt(MPTCP_ENABLE_APP_KEY)

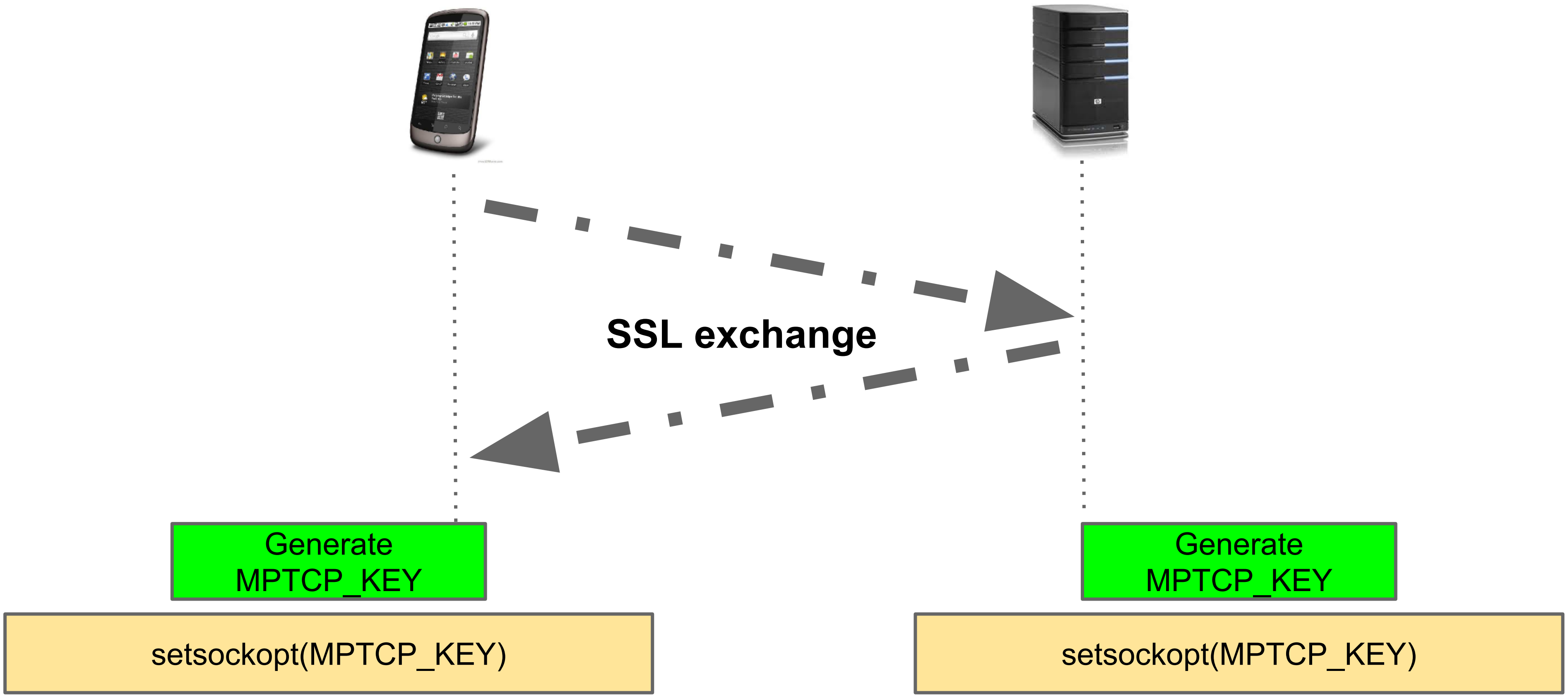
SYN

MP_CAPABLE
B-bit set

SYN+ACK

MP_CAPABLE
B-bit set

SSL initial handshake



SSL exchange

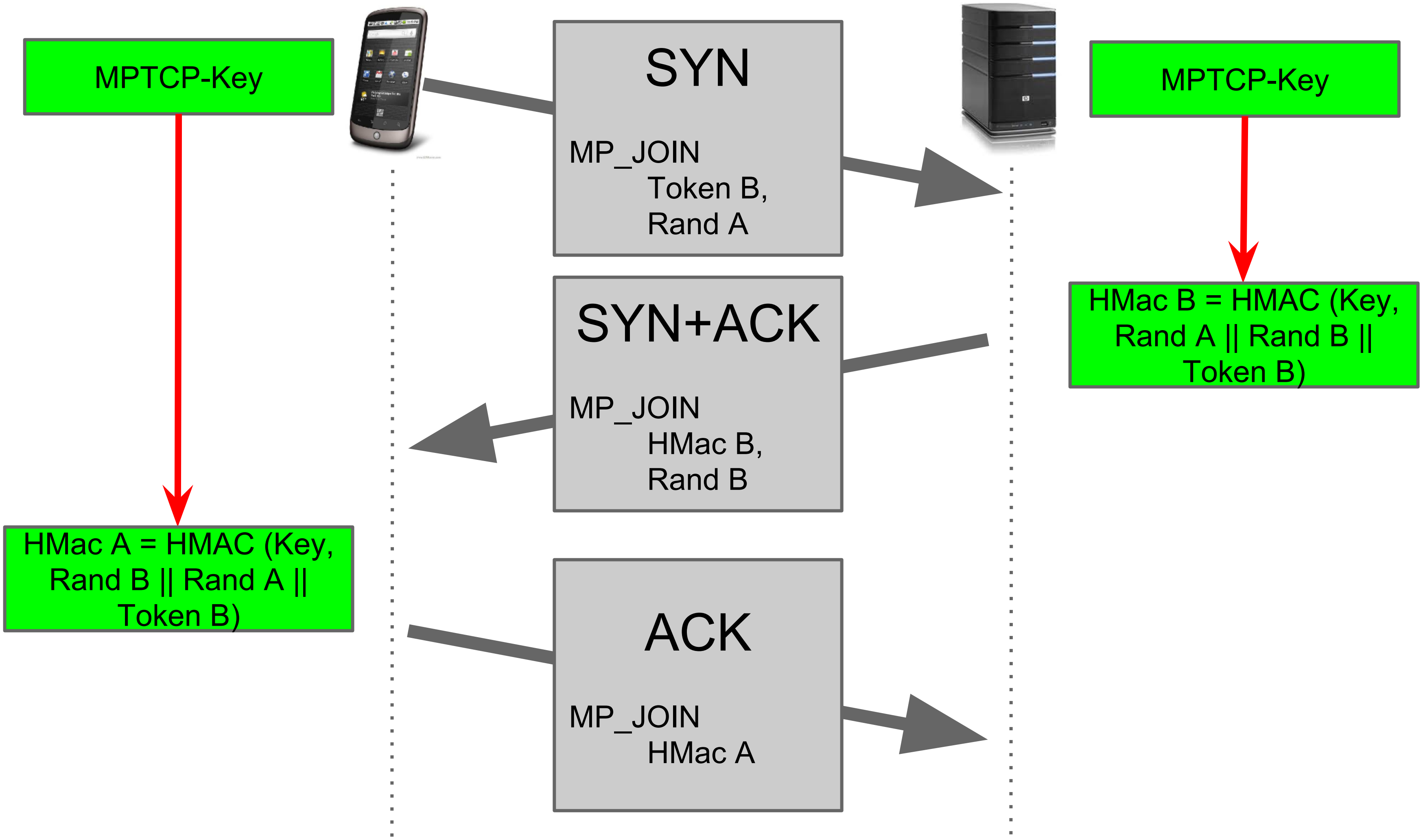
Generate MPTCP_KEY

setsockopt(MPTCP_KEY)

Generate MPTCP_KEY

setsockopt(MPTCP_KEY)

SSL additional subflow



MPTCP-Key

SYN

MP_JOIN
Token B,
Rand A

MPTCP-Key

HMac B = HMAC (Key, Rand A || Rand B || Token B)

SYN+ACK

MP_JOIN
HMac B,
Rand B

HMac A = HMAC (Key, Rand B || Rand A || Token B)

ACK

MP_JOIN
HMac A

Conclusion

- Application-level encryption instead of TCPcrypt
- Use the application’s key for MPTCP

Worth pursuing?