TCP Extended Data Offset Option

draft-touch-tcpm-tcp-edo-03
IETF 90 - Toronto

Joe Touch, USC/ISI

Overview

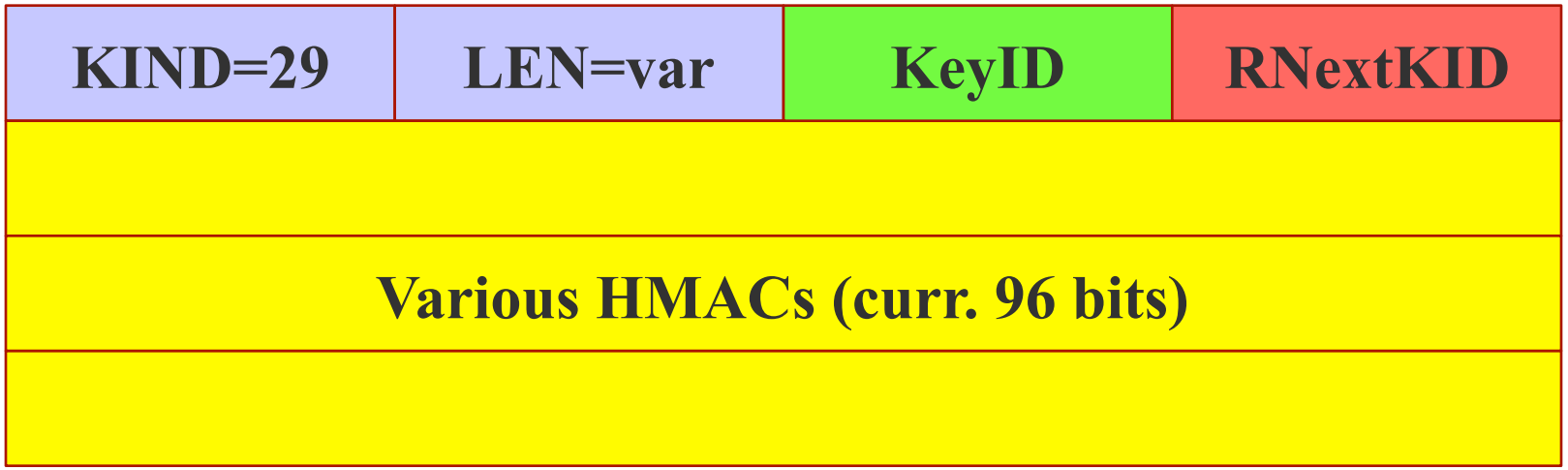
- Extend TCP-AO to encrypt
  - Simple extension to AO Master Key
  - Simple extension to AO segment processing
  - Inherits the rest of AO
- Two modes
  - Per-connection, in-band initial key exchange
  - AO-liked OOB Master Key configuration

Extension to TCP AO

- Extend TCP-AO to encrypt
  - Master Key Tuple adds:
    - Flag: encrypt (in addition to integrity check)
    - Flag: BTNS vs. AUTH mode
    - Symmetric traffic encryption key
      - Directly exchanged or MKT-derived
  - Processing:
    - Encrypt payload in-place using traffic encryption key
    - Authenticate payload using AO

TCP-AO

- Header includes KeyIDs:



- KeyID = current key for this HMAC
- RNextKeyID = “ready” to receive key ID

TCP-AO Features

- Loss-free rekeying – KeyID
- Loss-free sync. use of new key – RNextKID
- Per-connection keys
  - Master key + ISNs -> conn. Key
- Replay protection
  - Via sequence number extensions
- Master key tuple (MKT) includes parameters
  - Conn. key alg., HMAC alg., TCP option incl. flag
- Fully specified w.r.t. TCP states/events

TCP-AO Summary

Algorithm agile	**MKT indicates alg. Algs. specified separately.**
Allows rekeying, esp. efficiently	**KeyID for current segment MKT. RNextID for return path sync.**
Replay protection	**Ext. sequence numbers maintained, used in HMAC.**
Per-connection keys	**Derived keys using KDF.**
Man/auto KMI agnostic	**MKT treated as external. Parameter changes require MKT changes.**

Two AO-ENC modes

- BTNS-mode
  - Diffie-Hellman public keys in SYN, SYN/ACK
    - Unprotected segments (SHOULD NOT include data)
    - As with AO, limited to connections that match MKT
  - Full protection thereafter
- AUTH-mode
  - Pre-deployed Master Key Tuples using out-of-band key management
    - Same as AO
    - Authenticates too if pre-deployed tuples are signed
  - SYN, SYN/ACK processed same as other segments
    - Full protection throughout

Automatic MKT config (new)

(not in current draft)

- Setup MKT using DH exchange
  - Unprotected connection to a known port
  - Can exchange larger DH public keys
- Run in AUTH-mode thereafter
  - Per-connection traffic encryption keys derived using AO rules

Comparison to TCPINC Req'ts

• Requirements	
– Works where TCP works (over NATs, BEHAVE-compliant boxes)	**YES, same as AO NAT**
– Usable by unmodified apps	**YES**
– Usable with modified apps	**YES**
– Crypto algorithm agility	**YES, same as AO**
– Graceful fallback	**YES, same as AO**
– Encryption protects against passive eavesdropping	**YES**
– Minimize TCP option space, esp. SYN	**YES, same as AO**
– No required auth/config app/user interaction,	**YES, in BTNS mode**
– User/app Interaction possible	**YES, in AUTH mode**
– Good performance/latency (e.g., caching to avoid public key ops)	**YES**
– When encryption is enabled:	
• Provides forward secrecy	**YES**
• Integrity protects payload at least	**YES, incl. TCP/IP header**
• Encrypts payload	**YES**
• No extra 'linkability'	**YES, same as AO**
• Avoids enhancing fingerprinting	**YES, same as AO**
• Features not listed above	
– Per-connection granularity	**YES, same as AO**
– Key rollover without significant TCP impact	**YES, same as AO**
– Lower overhead vs. stacked protocols	**YES, same as AO**

Issues

- Diffie-Hellman public key length
  - DH expects 1024b (128B, not b), AO uses 96 bit HMAC
  - Truncate? Use a slightly larger field (192b)? (may need SYN-EOS)
  - Exchange OOB to avoid size limits?
- Key rollover
  - Current doc disables rollover in BTNS, assumes OOB update in AUTH
  - Add additional DH public key exchanges? (may need EDO)
- Specific KDF, E-KDF, and encryption algs
  - Spec'd in a separate doc as was done with AO
  - Can be decided by a separate group IMO
- TCP Option Number
  - Not strictly needed; can consider this a mode of AO