TCP-ENO: Encryption Negotiation Option

draft-ietf-tcpinc-tcpeno

Andrea Bittau, Dan Boneh, Daniel Giffin, Mark Handley,
David Mazières, and Eric Smith

IETF96

Tuesday, July 18, 2016

Goal

Abstract away details of TCPINC encryption protocols

Facilitate adoption of future TCP-level encryption specs

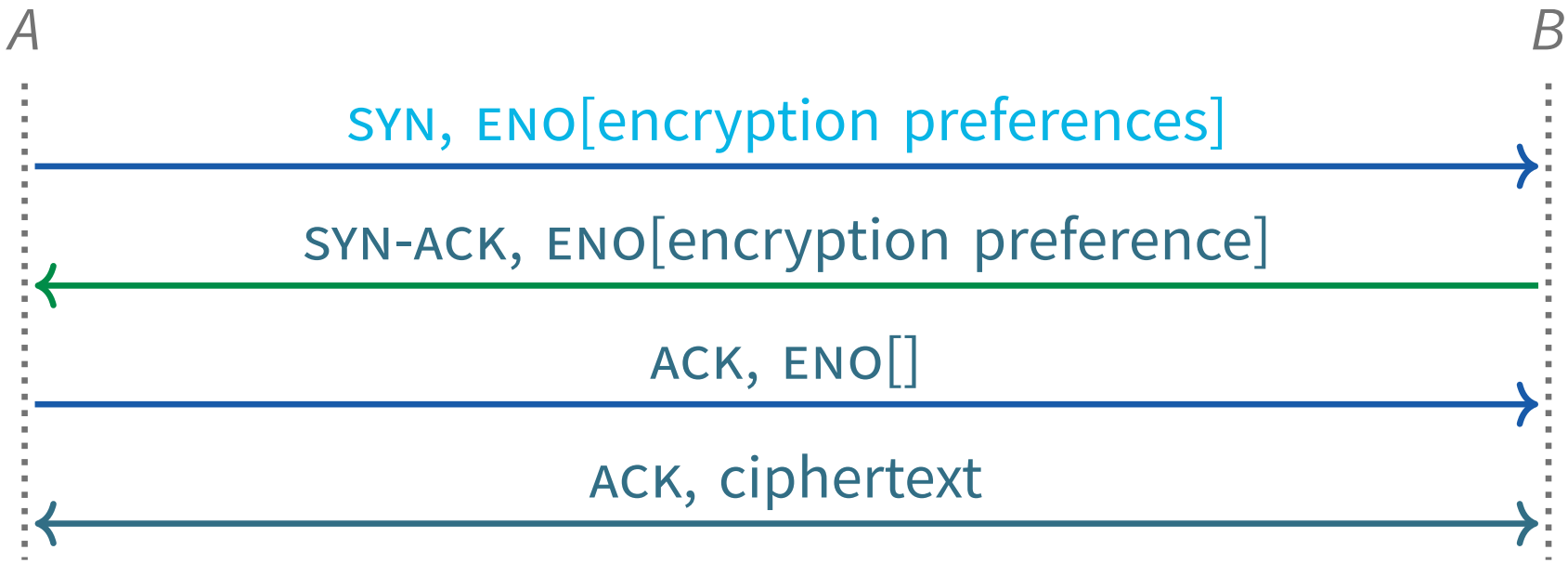
- New specs do not require additional TCP option kinds
- New specs incrementally deployable, fall back to older specs
- New specs compatibile with existing TCPINC-aware applications (recall charter requires authentication hooks)

Minimize consumption of TCP option space

Avoid unnecessary round trips for connection setup

Revert to unencrypted TCP when encryption not possible

Overview of common case



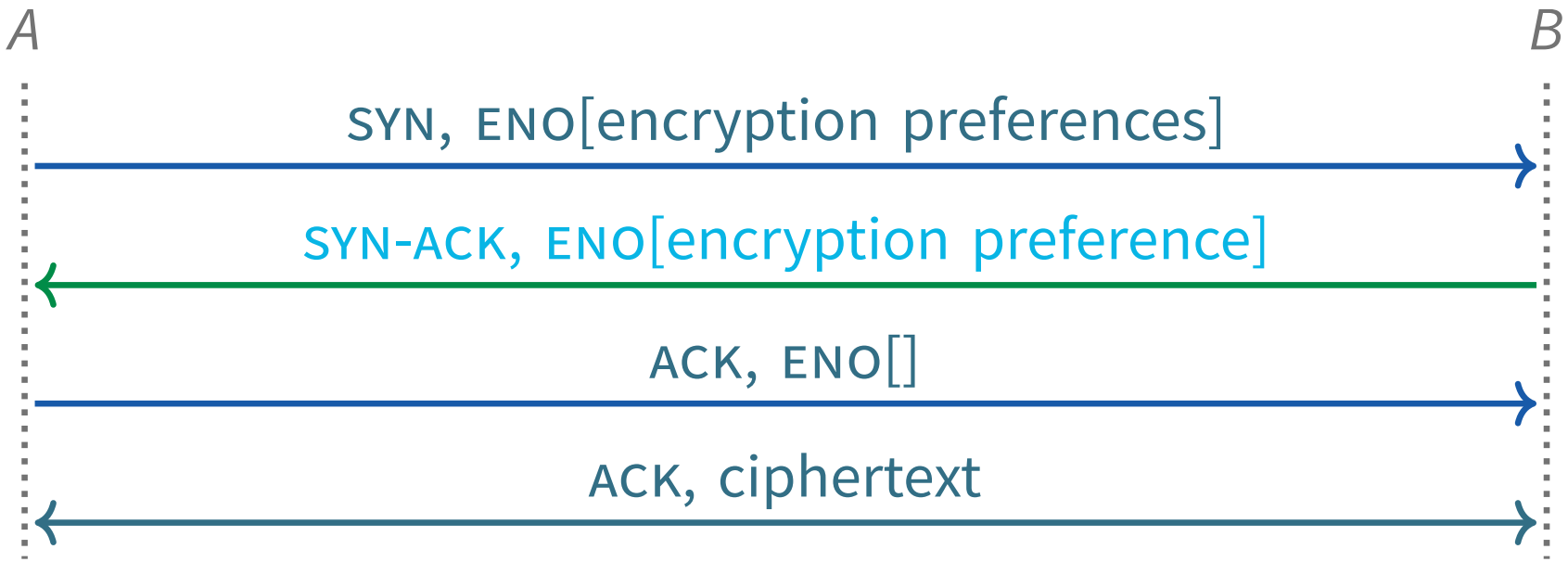
Active opener *A* lists spec preferences in ENO option

Passive opener *B* lists spec preferences in ENO option

***A* sends empty ENO option indicating encryption enabled**

If any of the above ENOs missing, revert to unencrypted TCP

Overview of common case



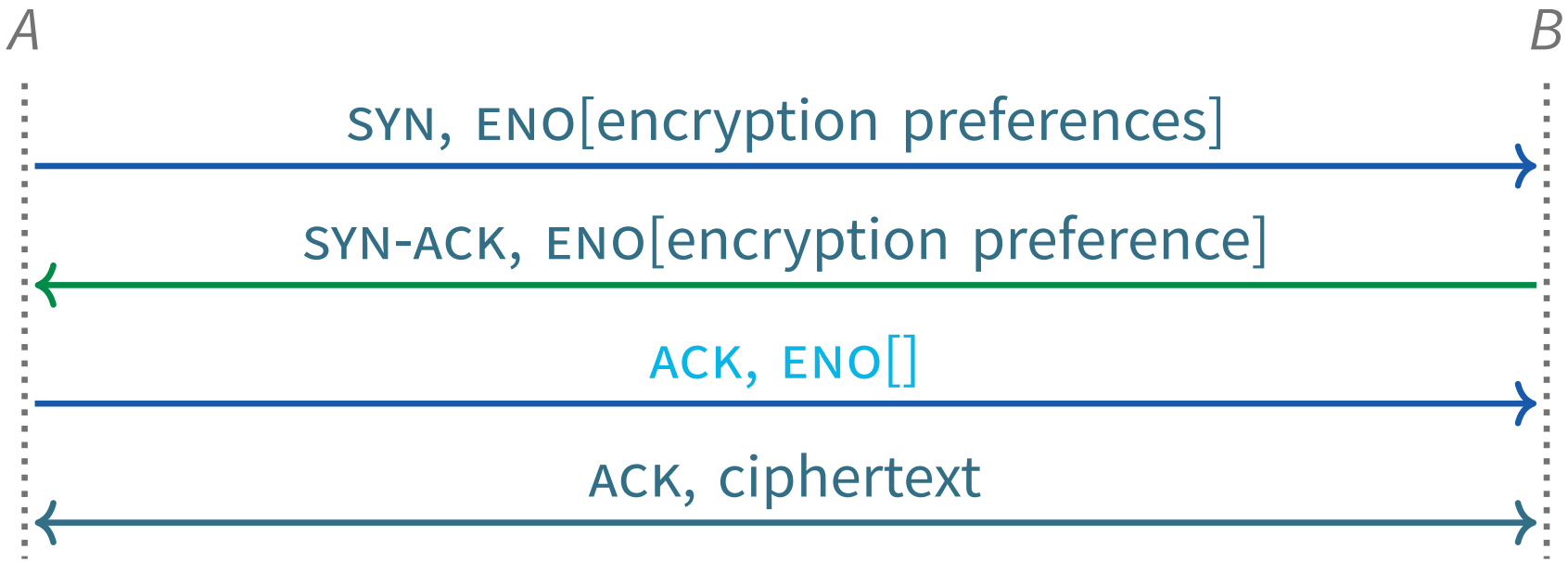
Active opener *A* lists spec preferences in ENO option

Passive opener *B* lists spec preferences in ENO option

***A* sends empty ENO option indicating encryption enabled**

If any of the above ENOs missing, revert to unencrypted TCP

Overview of common case



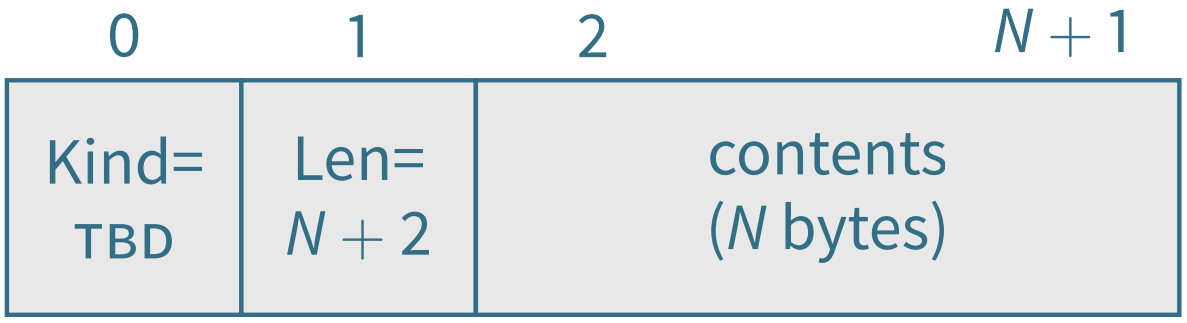
Active opener *A* lists spec preferences in ENO option

Passive opener *B* lists spec preferences in ENO option

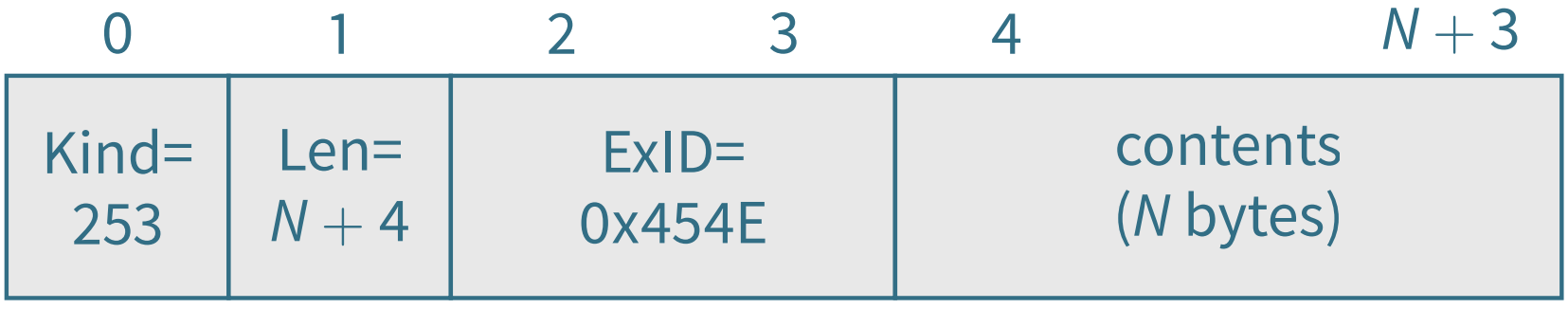
***A* sends empty ENO option indicating encryption enabled**

If any of the above ENOs missing, revert to unencrypted TCP

Two kinds of ENO option



MUST not use before TBD assigned by IANA



(New)

The good news: we officially have RFC6994 ExID 0x454E

- Current implementation now fully IANA compliant

The bad news: extra two bytes may push us over the edge

- E.g., tcpcrypt session resumption and default OSX options use all 40 bytes of option space in SYN segment

ENO option contents in SYN segments

SYN-form ENO is a container for a set of *suboptions*

Zero or one *general* suboption

One or more *spec identifier* suboptions

- Lists supported encryption specs
- Host *B* (passive opener) SHOULD list only one spec if possible
- Otherwise, *B* lists in order of increasing preference

ENO contents in non-SYN segments

ignored by ENO

Non-SYN-form ENO is just a flag (present/not present)

- Required for graceful fallback when ENO stripped from SYN-ACK

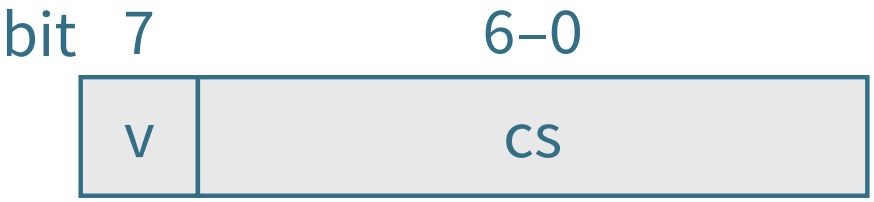
Contents does not matter

- Available for use by encryption specs
- If negotiated spec does not specify use, SHOULD be 0 bytes

New: Send in all segments until you receive non-SYN segment

- Wasn't totally clear in draft
- Required to recover from lost initial ACK segment

Initial suboption byte

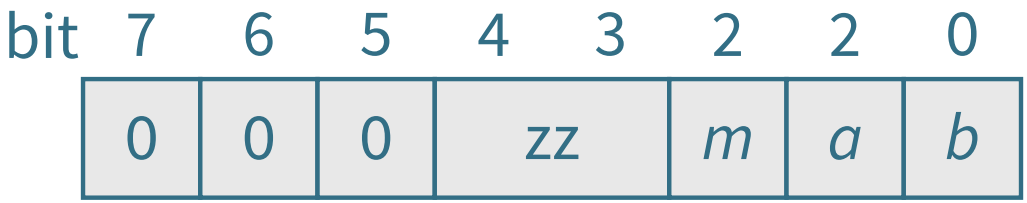


cs	v	meaning
`0x00-0x1f`	`0`	General suboption
`0x00-0x1f`	`1`	Length field
`0x20-0x7f`	`0`	Spec identifier suboption without data
`0x20-0x7f`	`1`	Spec identifier suboption followed by data

New: Eliminated reserved values

- General suboption now five bits
- Makes table easier to understand

General suboption format



***b* – Passive role bit**

- New: Required to be 1 for all passive openers
- New: Disable ENO if both sides have same value (eliminated *p* bit)

***a* – Application-aware bit (New: one bit, not two)**

- Mandatory application aware doesn't require separate bit

***m* – New: Middleware bit**

- Similar to *a*, but for use by cross-application middleware

zz – Reserved (send as 00 and ignore on receipt)

New: Ignore all but first general suboption in ENO

- If necessary, can later define bits in second 0x00–0x1f byte

Why both *a* and *m* bits?

Previously had two *a* bits (so *m* doesn't consume another bit)

***a* bit negotiates changes in application protocol**

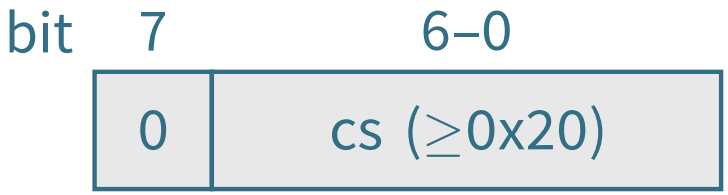
- Example: hash role+session ID into authentication cookie
- Intent: no future drafts place any cross-application restrictions on use

***m* bit negotiates authentication protocol before application**

- Negotiated protocol happens entirely before application protocol
- Future draft required to provide guidance
  - Need authentication protocol id or GUID to multiplex bit
  - Maybe length field to skip unsupported authentication messages
- Example: Sign session ID with public key in DNSSEC
  - Assumes DANE + secure DNS record saying server supports protocol
- Intent: secure legacy applications with `LD_PRELOAD`/shared lib upgrade

Spec identifier suboption format

Single-byte spec identifier suboption



- Indicates support for spec cs:

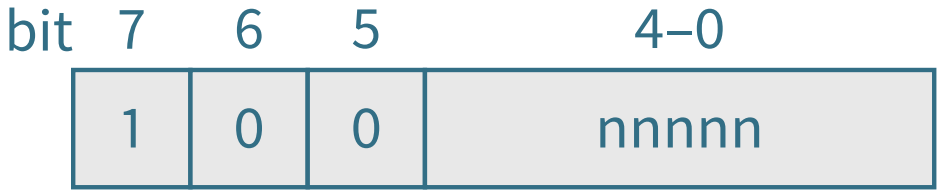
Spec identifier suboption with suboption data

- Indicates support for spec cs
- Format and meaning of byte determined by spec cs

Spec identifier suboption length

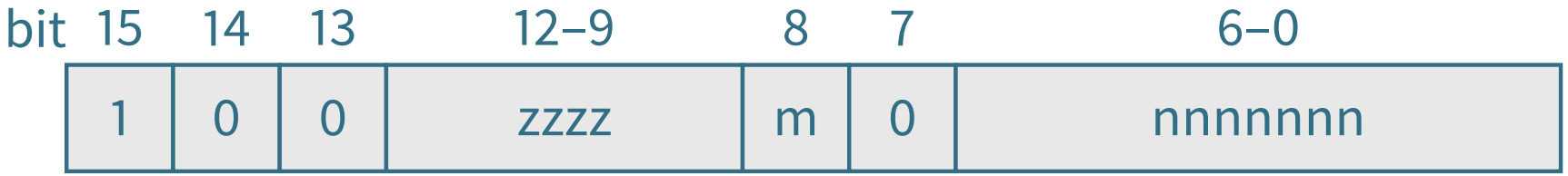
By default, a multi-byte suboption extends to end of TCP option

Alternatively, can be preceded by length byte



- Indicates suboption data length of nnnn+1 bytes (not counting spec id)

Or length word



- **nnnnnnn** – low seven bits of (length − 1)
- **m** – most significant bit of (length − 1)
- **zzzz** – New: must be 0 or disable ENO

New: disable ENO in all other cases

Other changes

Much document restructuring

- New terminology and rationale sections
- Be clearer about normative/non-normative sections

First attempt at plausible IANA considerations section

- Spec ID registry is *Specification Required*
- Spec IDs should be allocated when *Designated Expert* believes RFC is more likely than not
- Requests published to TCPINC or successor WG mailing list
- cs $=$ 0x20 reserved for experimental use

Improved security considerations

- Be sterner about opportunistic encryption and randomness

Pared down experiments section

Summary of major changes

Incorporate new ExID

Role negotiation requires different b bits at each end

- Passive openers must include general suboption with $b = 1$

General suboption now 5 bits

- Future extensibility from second 0x00-0x1f byte, not reserved values
- One application-aware (a) bit instead of two, but added m bit

256 byte maximum suboption data length

- Generally more precision on rejecting illegal SYN-form ENO options

Document improvements

Still to do

Agree on term “spec” or something better

Decide whether some RFC5705-like exporter mechanism needed

- Existing requirements sufficient to virtualize session ID

Maybe take measures to free up SYN option space

- (Flame-bait) Dedicate one bit of general suboption to declaring timestamp supported, saving 10 bytes of SYN option space…
- Get dedicated TCP option (preferably `'E'` – 69)

Ideally not too much else before RFC

Work needed for follow-on/companion documents:

- TCP-ENO middlebox probing
- How to multiplex experimental spec ID 0x20 (ExID-like mechanism)
- How to multiplex the middleware *m* bit (some length/uuid protocol)