Bytes to Schlep? Use a FEP:
Hiding Protocol Metadata with Fully Encrypted Protocols

Ellis Fenske (U.S. Naval Academy)

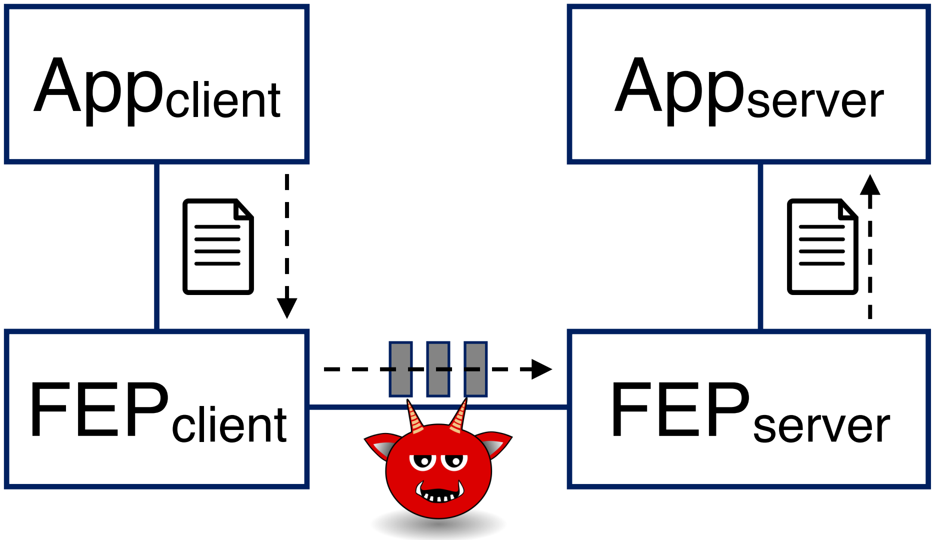
Aaron Johnson (U.S. Naval Research Laboratory)

May 26th, 2024

Cryptographic Applications Workshop (CAW 2024)

Fully Encrypted Protocols (FEPs)

What is a Fully Encrypted Protocol (FEP)?



1. All bytes look random
2. Message lengths variable

Real-world examples:

- obfs4 / lyrebird (Tor)
- shadowsocks (Outline VPN)
- Obfuscated SSH (Psiphon)
- OpenVPN + XOR patch
- Vmess (V2Ray)

Summary of Our Work

Problem: No precise understanding of FEPs
- Goals not formalized mathematically
- Security cannot be proven
- Existing FEPs continually present security flaws
- IND$-CPA: similar goal but for atomic messaging

Solutions:
1. New security definitions for FEPs
2. Relations among new and existing security definitions
3. Secure constructions of FEPs
4. Analysis of existing FEPs

Status of this Work

- Presented early version of this work at FOCI 2023
  - Future Work from that talk:
    1. Proving security of our construction
    2. Deriving relations between the security definitions
    3. Addressing forward secrecy via key exchange in the protocol
    4. Extending our definitions to the datagram setting

Status of this Work

- Presented early version of this work at FOCI 2023
  - Future Work from that talk:

1. ~~Proving security of our construction~~
2. ~~Deriving relations between the security definitions~~
3. Addressing forward secrecy via key exchange in the protocol
4. ~~Extending our definitions to the datagram setting~~

U.S. NAVAL RESEARCH LABORATORY

Status of this Work

- Presented early version of this work at FOCI 2023
  - Future Work from that talk:
    1. ~~Proving security of our construction~~
    2. ~~Deriving relations between the security definitions~~
    3. Addressing forward secrecy via key exchange in the protocol
    4. ~~Extending our definitions to the datagram setting~~
  - Added experimental analysis of existing FEP security

Status of this Work

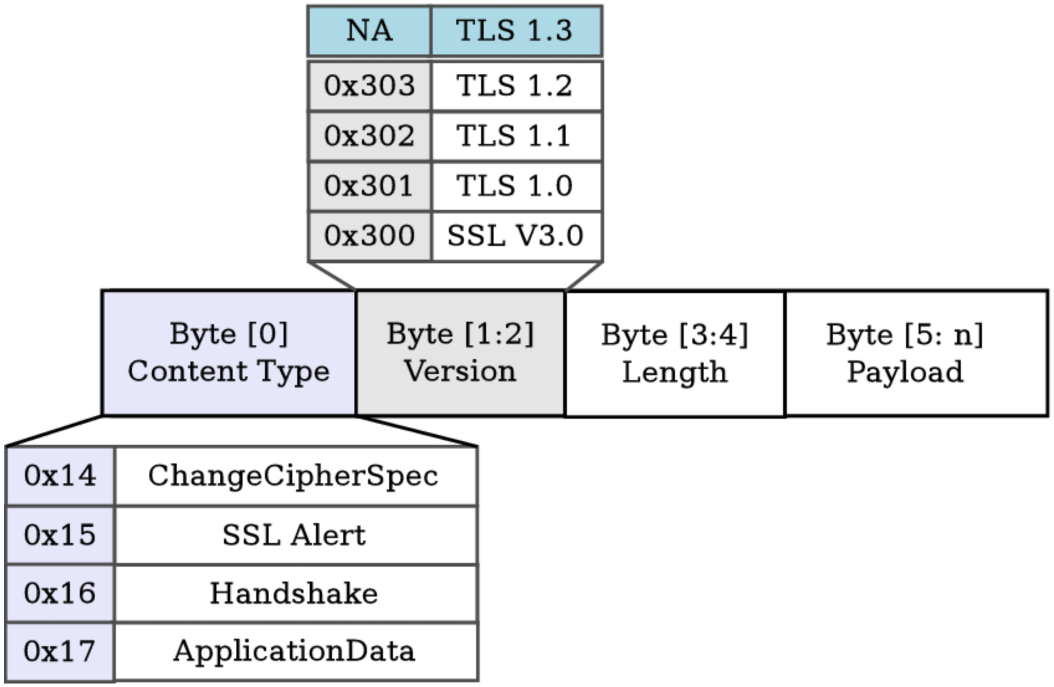
- Presented early version of this work at FOCI 2023
  - Future Work from that talk:
    1. ~~Proving security of our construction~~
    2. ~~Deriving relations between the security definitions~~
    3. Addressing forward secrecy via key exchange in the protocol
    4. ~~Extending our definitions to the datagram setting~~
  - Added experimental analysis of existing FEP security
- Paper available:
  - Ellis Fenske and Aaron Johnson. “Bytes to Schlep? Use a FEP: Hiding Protocol Metadata with Fully Encrypted Protocols”. May 2024.
  - <https://arxiv.org/abs/2405.13310>

Why FEP?

Existing encrypted protocols reveal metadata

- Protocol identity and version
- Amount of payload data
- Cryptographic primitives being used

Example 1: TLS Record

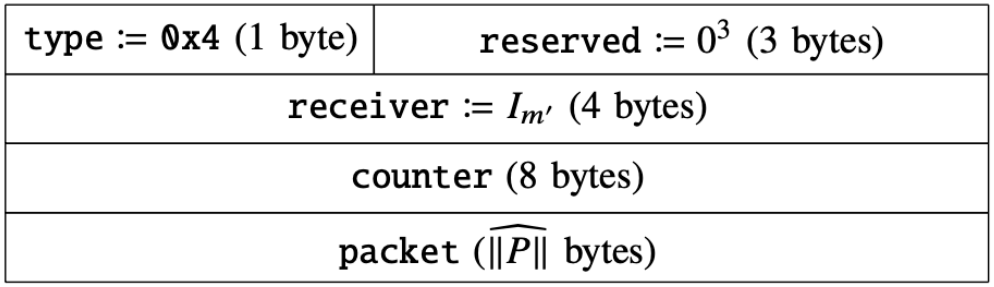


NA	TLS 1.3
0x303	TLS 1.2
0x302	TLS 1.1
0x301	TLS 1.0
0x300	SSL V3.0

Byte [0] Content Type	Byte [1:2] Version	Byte [3:4] Length	Byte [5: n] Payload

0x14	ChangeCipherSpec
0x15	SSL Alert
0x16	Handshake
0x17	ApplicationData

Example 2: WireGuard Datagram



`type` := `0x4` (1 byte)	`reserved` := 0^3 (3 bytes)
`receiver` := $I_{m'}$ (4 bytes)	
`counter` (8 bytes)	
`packet` ($\widehat{\|P\|}$ bytes)	

Why FEP?

FEP Reason #1: Censorship circumvention

- Typical VPN protocols can easily be identified and blocked
  - e.g. OpenVPN, WireGuard, IPSec
  - Censors have blocked VPN protocols (e.g. China, Russia)
- FEPs have been invented multiple times to eliminate simple protocol fingerprints (e.g. obfs4, shadow socks, Obfuscated SSH, Vmess)
- China has blocked FEPs: Wu et al. "How the Great Firewall of China Detects and Blocks Fully Encrypted Traffic". USENIX Security 2023.

Why FEP?

FEP Reason #2: Maximally protects metadata

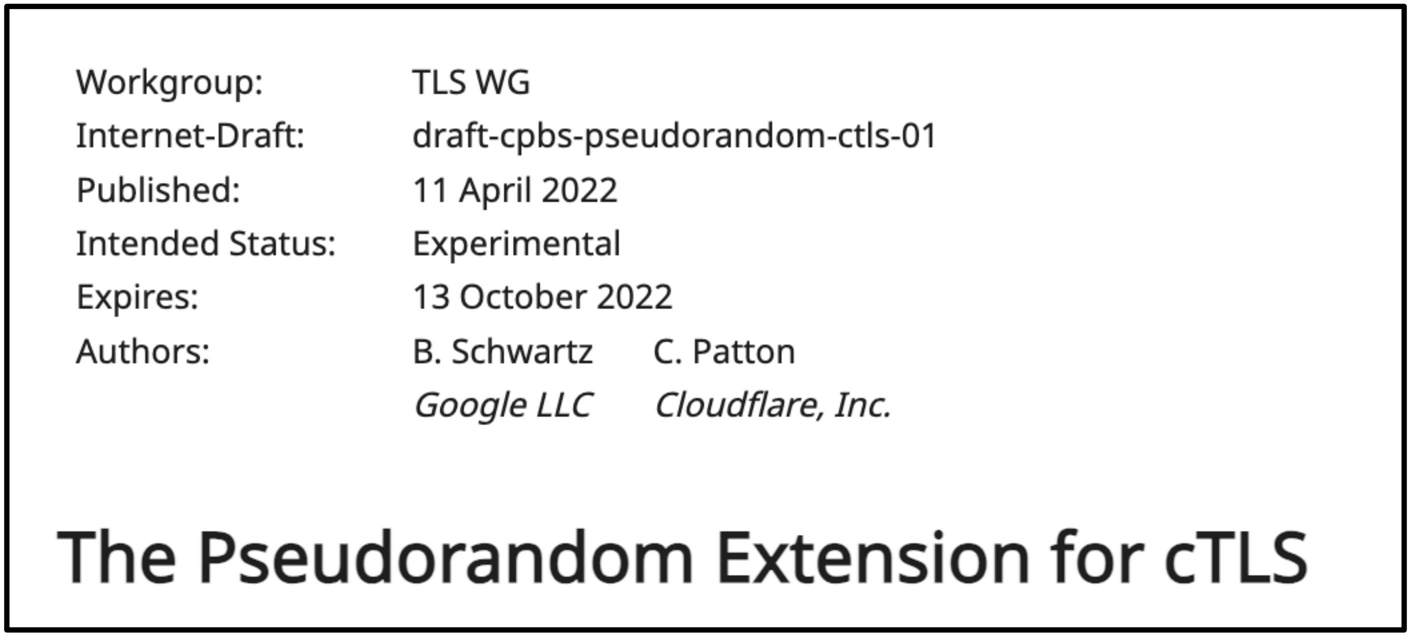
- Protocols increasingly protect metadata
  - QUIC
  - TLS 1.3 Encrypted Client Hello
  - Cryptocurrencies (Ethereum's RPLx, Lightning's Bolt)
- Metadata can be sensitive
  - Application(e.g. application-specific protocols)
  - Domain of the destination (e.g. SNI TLS extension)
  - Ciphertext primitives in use (some might be vulnerable)

Why FEP?

FEP Reason #3: Prevents Internet ossification

- Middleboxes develop around observable protocol features
  - Security firewalls
  - Traffic shapers
- Alternate solution: David Benjamin. 2020. RFC 8701 Applying Generate Random Extensions And Sustain Extensibility (GREASE) to TLS Extensibility

Why FEP?



Workgroup: TLS WG
Internet-Draft: draft-cpbs-pseudorandom-ctls-01
Published: 11 April 2022
Intended Status: Experimental
Expires: 13 October 2022
Authors: B. Schwartz, *Google LLC*; C. Patton, *Cloudflare, Inc.*

The Pseudorandom Extension for cTLS

- “**Privacy**: A third party… cannot tell whether two connections are using the same pseudorandom cTLS template”
- “**Ossification risk**”
- “**TODO**: More precise security properties and security proof. The goal we're after hasn't been widely considered in the literature so far, at least as far as we can tell.”

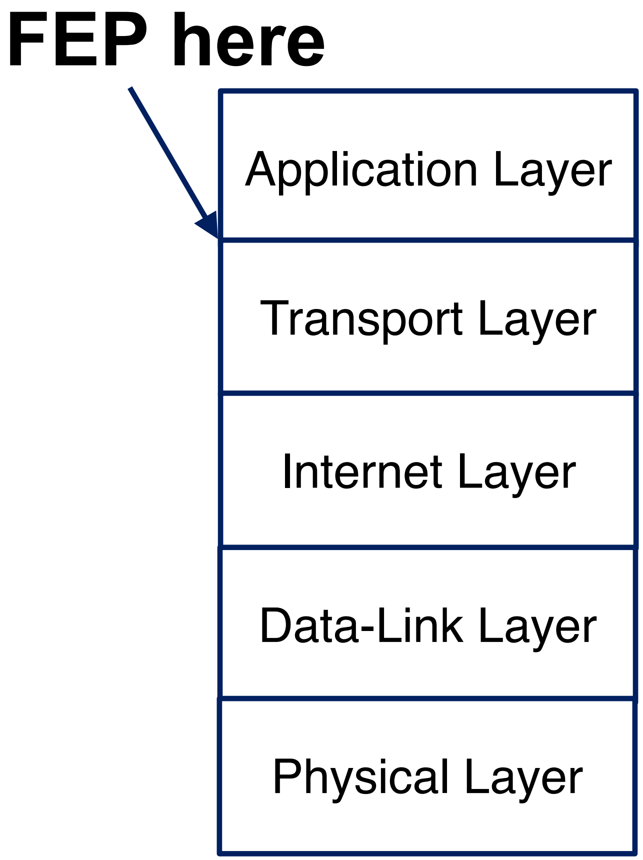
Encrypted Protocols

Non-FEP encrypted protocols innovation is still occurring:

- OSCORE: IoT-optimized (2019)
- NoiseSocket: generic framework (2017)
- WireGuard: VPN (2017)
- Bolt: Lightning network (2016)
- RLPx: Ethereum (2015)

Why couldn’t these all be FEPs?

FEPs in the Network Stack



Generally assume over TCP or UDP

- Below transport layer limits developer agility
  - Requires permissions for raw-socket access (e.g. iOS jailbreak)
- TCP and UDP are the common transport protocols
  - New reliable transports over UDP
    - e.g. QUIC, kcp
    - Difficult to accomplish while protecting metadata
- FEP terms
  - *Datastream FEP (e.g.* FEP over TCP)
  - *Datagram FEP* (e.g. FEP over UDP)

U.S. NAVAL RESEARCH LABORATORY

Looking at a FEP: obfs4

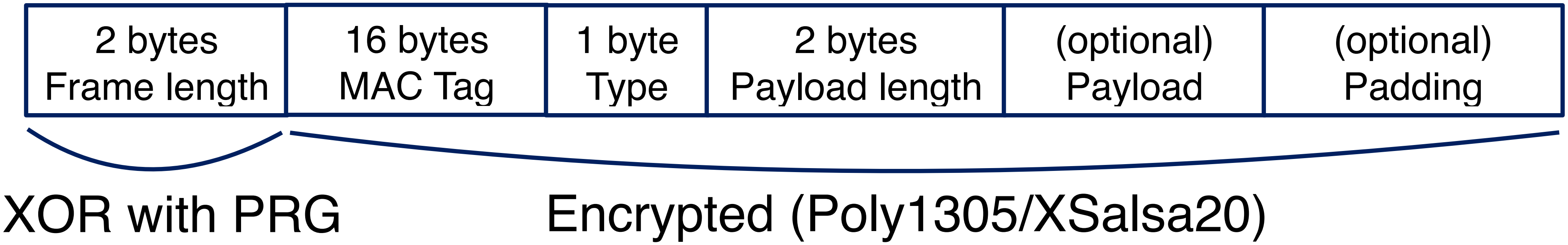
Tor’s obfs4 (aka lyrebird) is a sophisticated FEP

  - Uses TCP
  - Key exchange for forward secrecy
  - Padding for message-length variation

- Handshake

  1. Client sends: Elligator-encoded key + random padding
  2. Server sends: Elligator-encoded key + random padding

- Data-phase messages

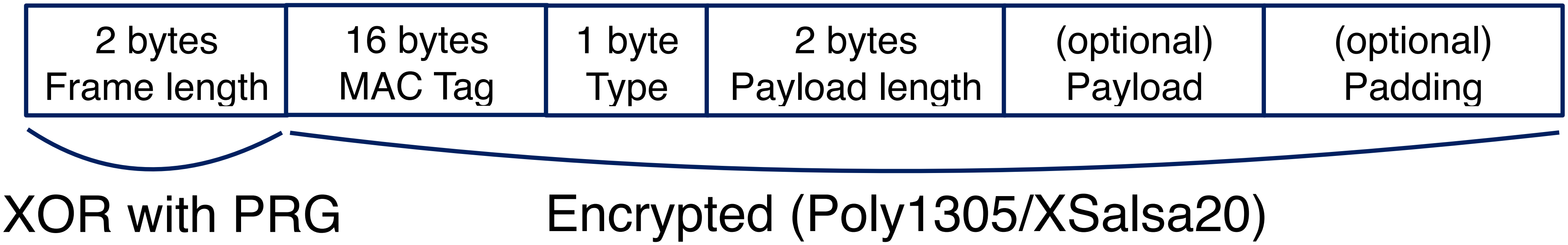


2 bytes Frame length	16 bytes MAC Tag	1 byte Type	2 bytes Payload length	(optional) Payload	(optional) Padding

XOR with PRG

Encrypted (Poly1305/XSalsa20)

Looking at a FEP: obfs4



2 bytes Frame length	16 bytes MAC Tag	1 byte Type	2 bytes Payload length	(optional) Payload	(optional) Padding

XOR with PRG

Encrypted (Poly1305/XSalsa20)

Security issues

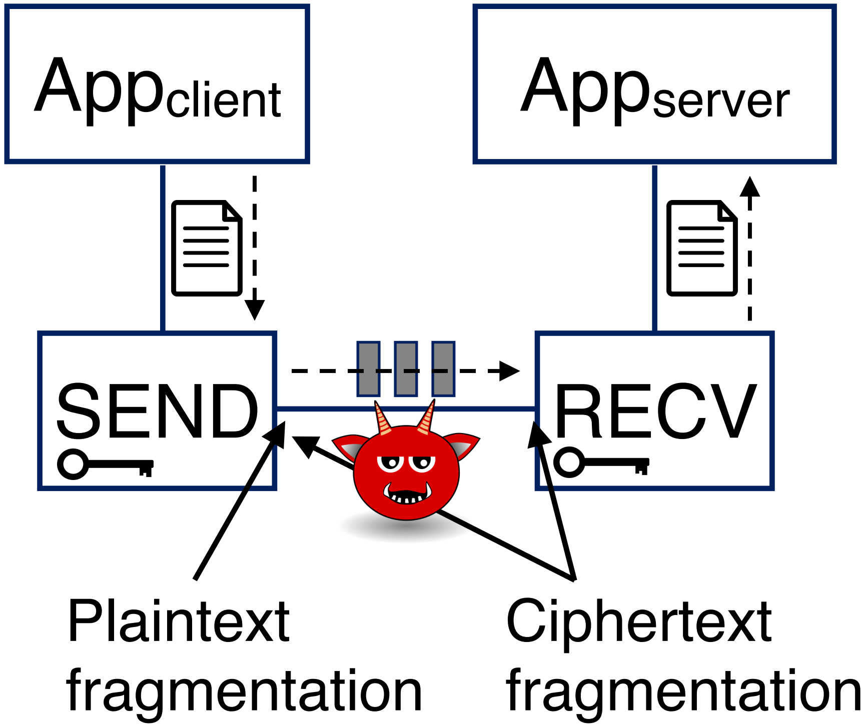
1. Length field is malleable
2. obfs4 closes connection upon decryption error
3. #1 + #2 = active attack reveals obfs4 message structure
4. Specific minimum message length despite padding

Let’s define FEP security to rule out such issues.

New FEP Security Definitions

1. Passive security:
    a. Datastream: **FEP-CPFA**
       (FEP under Chosen Plaintext-Fragment Attacks)
    b. Datagram: **FEP-CPA**
       (FEP under Chosen Plaintext Attacks)
2. Active security:
    a. Datastream: **FEP-CCFA**
       (FEP under Chosen Ciphertext-Fragment Attacks)
    b. Datagram: **FEP-CCA**
       (FEP under Chosen Ciphertext Attacks)
3. Message sizes: **Traffic Shaping**

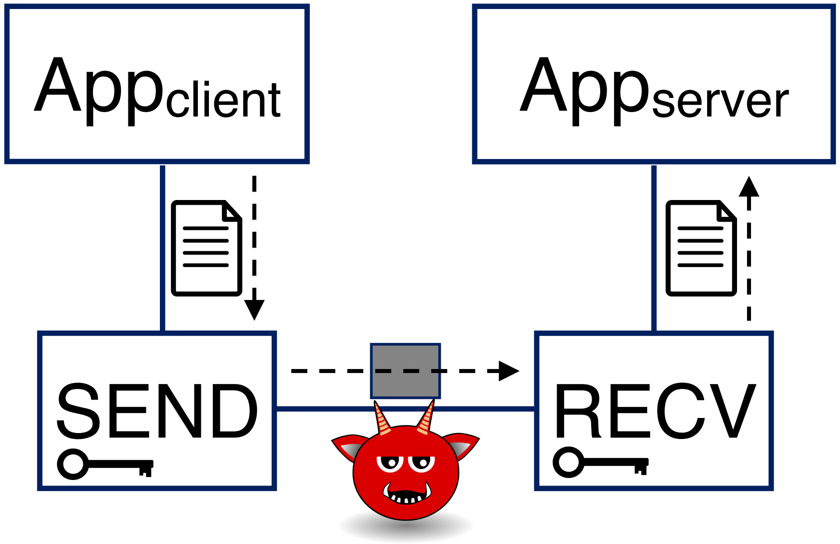
Datastream Setting



- Unidirectional channel
- Model allows pre-shared state
- Datastream semantics*
  - Inputs and outputs treated as byte streams
  - Reliable, in-order delivery
  - Models TCP

*Marc Fischlin, Felix Günther, Giorgia Azzurra Marson, and Kenneth G. Paterson. "Data is a stream: Security of stream-based channels". CRYPTO 2015.

Datagram Setting



- Unidirectional channel
- Model allows pre-shared state
- Datagram semantics*
  - Inputs and outputs treated as atomic messages
  - Messages may be dropped or reordered
  - Models UDP

*Similar to: Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. “Authenticated encryption in SSH: provably fixing the SSH binary packet protocol”. ACM CCS 2002.

Protocol Model

SEND

Input

- m : plaintext message
- p : packet length
- f : flush flag (datastream)

Output

- c : ciphertext

RECV

Input

- c : ciphertext

Output

- m : plaintext message
- C : channel close flag (datastream)

In implementation, SEND and RECV would interact with *sockets*.

Passive FEP Security (datagram and datastream)

Security experiment

1. Challenger chooses bit *b*.
2. Adversary can query stateful oracle O^b_{SEND}.
3. Adversary outputs guess *b*'.
4. Success if *b*'=*b*.

Definition: *Protocol is passively FEP secure if advantage over random guessing is negligible.*

Real World

$O^0_{\text{SEND}}(m,p,[f])$

- Outputs SEND(m,p,[f])

Random World

$O^1_{\text{SEND}}(m,p,[f])$

- Outputs |SEND(m,p,[f])| random bytes

Active security (datastream): FEP-CCFA (Chosen Ciphertext-Fragment Attacks)

Security experiment

- CLOSE($||C_S$, C_R): Secure close function
- $||C_S$: concatenated O^b_{SEND} outputs
- C_R: O^b_{RECV} inputs

1. Challenger chooses bit *b*.
2. Adversary can query stateful oracles O^b_{SEND} and O^b_{RECV}.
3. Adversary outputs guess *b*'.
4. Success if *b*'=*b*.

Definition: *Protocol is FEP-CCFA if advantage over random guessing is negligible.*

Real World

$O^0_{\text{SEND}}(m,p,f)$

- Outputs SEND(m,p,f)

$O^0_{\text{RECV}}(c)$

- Always returns channel close flag *C*.
- Does not return output message *m* unless out of sync.

Random World

$O^1_{\text{SEND}}(m,p,f)$

- Outputs |SEND(m,p,f)| random bytes

$O^1_{\text{RECV}}(c)$

- Returns channel close flag CLOSE($||C_S$, C_R).
- Does not return output message *m*.

Active security (datagram): FEP-CCA (Chosen Ciphertext Attacks)

Security experiment

- *null* message output allowed to be ignored to enable short chaff messages w/o MAC

1. Challenger chooses bit b.
2. Adversary can query stateful oracles O^b_{SEND} and O^b_{RECV}.
3. Adversary outputs guess b'.
4. Success if $b'=b$.

Definition: *Protocol is FEP-CCA if advantage over random guessing is negligible.*

Real World

$O^0_{\text{SEND}}(m,p)$

- Outputs SEND(m,p)

$O^0_{\text{RECV}}(c)$

- Output m returned if:

1. c not Send output,
2. m not error, and
3. m not *null*

Random World

$O^1_{\text{SEND}}(m,p)$

- Outputs |SEND(m,p)| random bytes

$O^1_{\text{RECV}}(c)$

- Does not return output m.

Secure Close Functions

- Secure close function CLOSE($||C_S$, C_R)
  - $||C_S$: concatenated SEND outputs
  - C_R: RECV inputs
  - Ensures closures give no more information than network observations
    - E.g. No closure based on plaintext value
  - Rules out obfs4 behavior because length fields cannot be identified in concatenated byte sequence
- Examples of secure close functions
  - Never close (e.g. shadowsocks requests)
  - Close after timeout
  - Close at first “sync” byte position after modified byte

Traffic Shaping

Definition (datastream): *Protocol satisfies* Traffic Shaping *if, for all messages m and p ≥ 0,*

$|\mathrm{SEND}(m,p,f{=}0)| = p$, and

$|\mathrm{SEND}(m,p,f{=}1)| \geq p$.

Definition (datagram): *Protocol satisfies* Traffic Shaping *if, for all messages m and L≥p≥0, with c* ← SEND(m,p),

If c is not an error, then $|c| = p$, and

If m is null, then c is not an error.

- Enables arbitrary-length messages
- Generalizes padding functionality of existing FEPs
- Avoids protocol-specific minimum-message sizes

Other FEP security requirements*

- Confidentiality
  - IND-CCFA/IND-CCA (Datastream/Datagram)
  - Not implied by FEP-CCFA/CCA because ciphertext lengths can leak plaintexts
  - With length regularity, implied by FEP-CCFA/CCA
- Integrity
  - INT-CST/INT-CTXT (Datastream/Datagram)
  - Implied by FEP-CCFA/CCA

Experimental Analysis of Datastream FEPs

Datastream Protocol	Close Behavior	FEP-CPFA	FEP-CCFA	Length Obfuscation	Minimum Message Size
Shadowsocks-libev (request/response)	Never / Auth Fail	✅	✅ / ❌	None	35
V2Ray-Shadowsocks (request/response)	Drain / Auth Fail	✅	❌	None	35
V2Ray-VMess	Drain	✅	❌	Padding	18
Obfs4/Lyrebird	Auth Fail	✅	❌	Padding	44
OpenVPN-XOR	Auth Fail	❌	❌	None	42
Obfuscated-OpenSSH (-PSK)	Auth Fail	❌ (✅)	❌	None	16
kcptun	Never	✅	❌	None	52
Our construction	Never	✅	✅	Traffic Shaping	1

- Generally close behavior is identifying, even when they tried to avoid that
- Minimum message size may not appear in practice, although protocols with keepalives *do* generate them
- Our experiments uncovered an integrity attack in VMess (now fixed)

Experimental Analysis of Datagram FEPs

Datagram Protocol	FEP-CPA	FEP-CCA	Length Obfuscation	Minimum Message Size
Shadowsocks-libev	✅	✅	None	55
WireGuard-SWGP	✅	✅	Padding	75
OpenVPN-XOR	❌	❌	None	40
Our construction	✅	✅	Traffic Shaping	0

- FEP security easier to achieve without closures
- We observe larger minimum message size due to more required metadata in the datagram setting.

Future Work

- FEP research ideas
  - Forward secrecy
  - Forward metadata secrecy
  - High-performance FEPs
  - Other TCP metadata leaks (e.g. congestion window)
  - Versioning / protocol negotiation
- Paper available:
  - Ellis Fenske and Aaron Johnson. “Bytes to Schlep? Use a FEP: Hiding Protocol Metadata with Fully Encrypted Protocols”. May 2024.
  - <https://arxiv.org/abs/2405.13310>