MUFFLER: Secure Tor Traffic Obfuscation with Dynamic Connection Shuffling and Splitting

IEEE INFOCOM 2025

Minjae Seo[*], Myoungsung You[†], Jaehan Kim[†], Taejune Park[‡], Seungwon Shin[†], and Jinwoo Kim[§]

[*]ETRI, Republic of Korea

[†]KAIST, Republic of Korea

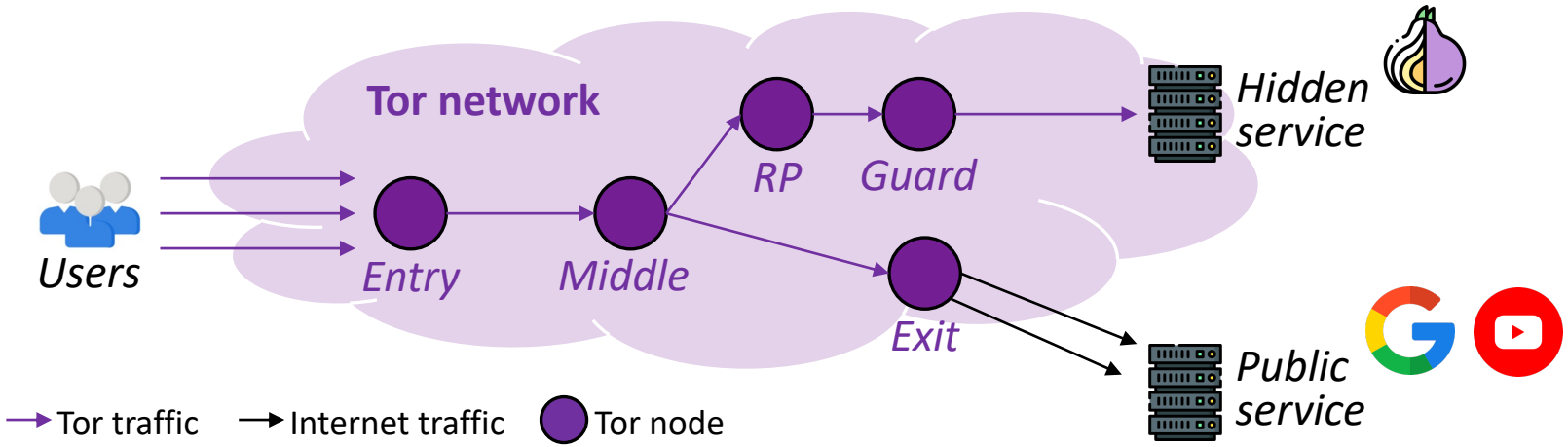
[‡]Chonnam National University, Republic of Korea

[§]Kwangwoon University, Republic of Korea

Minjae Seo and Myoungsung You contributed equally

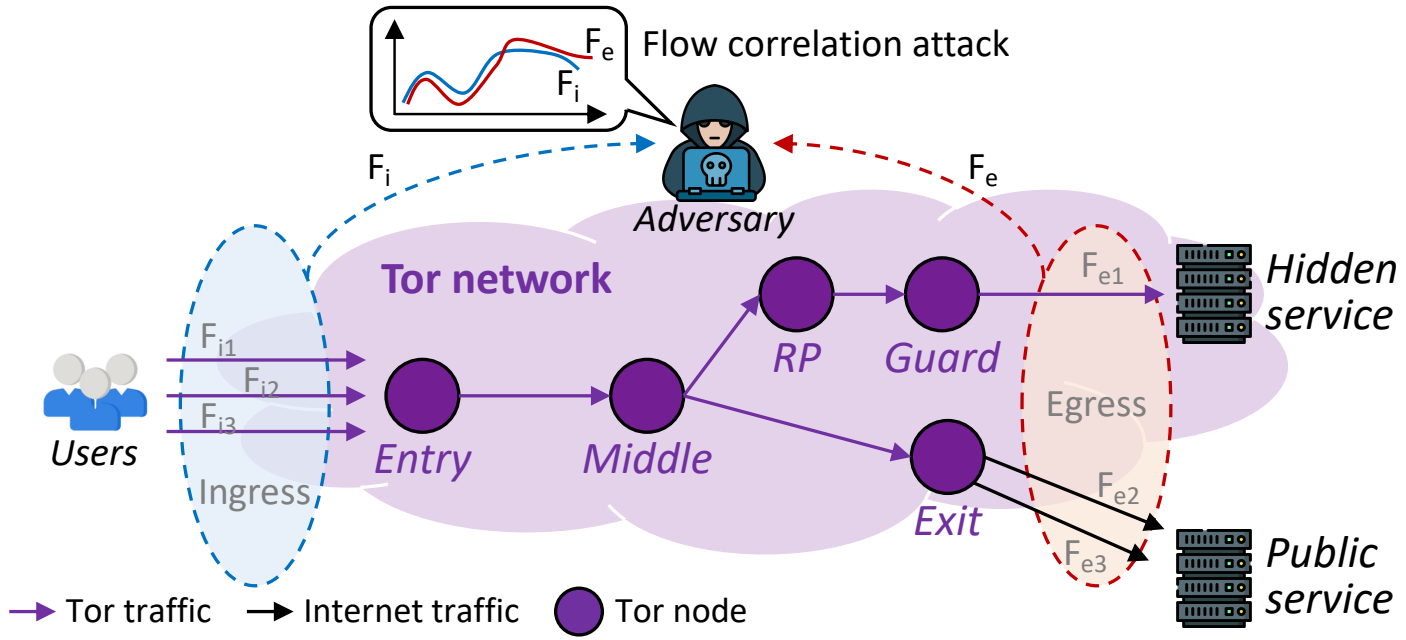
Tor

- Enhances privacy by routing traffic through ***volunteer nodes*** with encryption layers
  - I.e., Entry (guard), middle, exit nodes



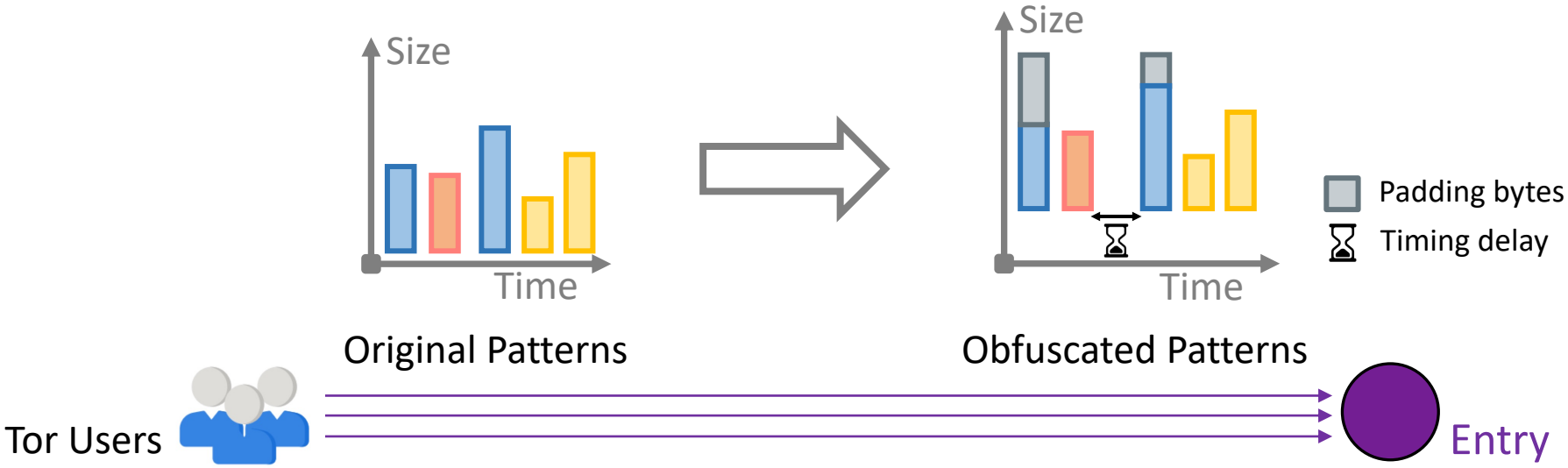
FCA (Flow Correlation Attack)

- Matches ***ingress/egress flows*** based on packet size and interval timings
- Links user IPs to destinations, **breaking anonymity** of Tor users
  - E.g., RAPTOR [USENIX SEC '15], DeepCorr [CCS '18], SUMo [NDSS '24]



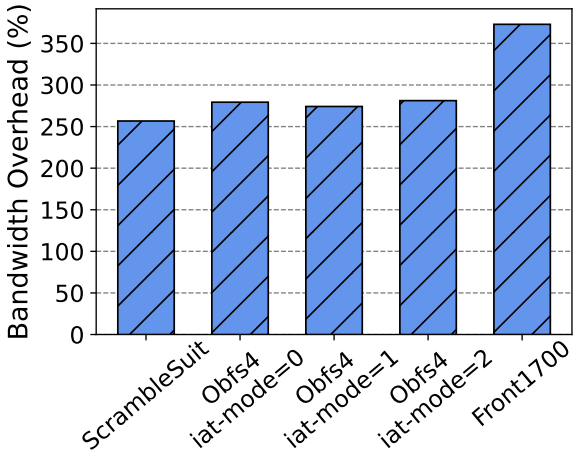
Existing Solution: Traffic Obfuscation

- Obfuscates Tor traffic patterns to mitigate FCAs
- Leverages ***padding bytes*** or ***inter-packet delays***
  - E.g., BuFLO [S&P ’12], WTF-PAD [ESORICS ’16], FRONT [USENIX SEC ’20]
- Often adopted by Tor as a client-side solution
  - E.g., ScrambleSuit, obfs4

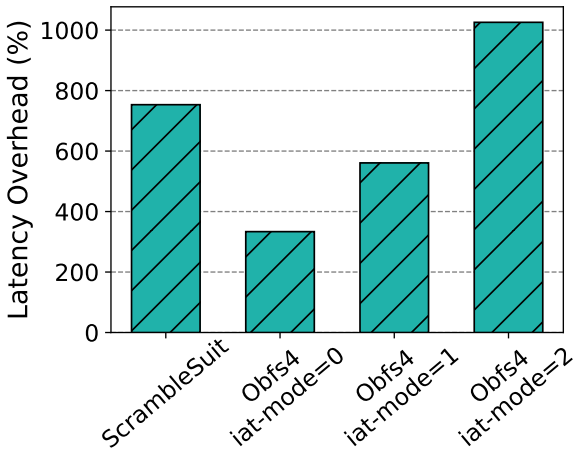


Limitation: Network Inefficiency

- Existing solutions incur high bandwidth and latency overhead
- Motivating example:
  - Conducted an experiment to obfuscate Tor traffic using ScrambleSuit, obfs4 (iat-modes 0/1/2), and Front 1700
  - Shows over 250% ***bandwidth overhead*** and up to 1000% ***latency overhead***



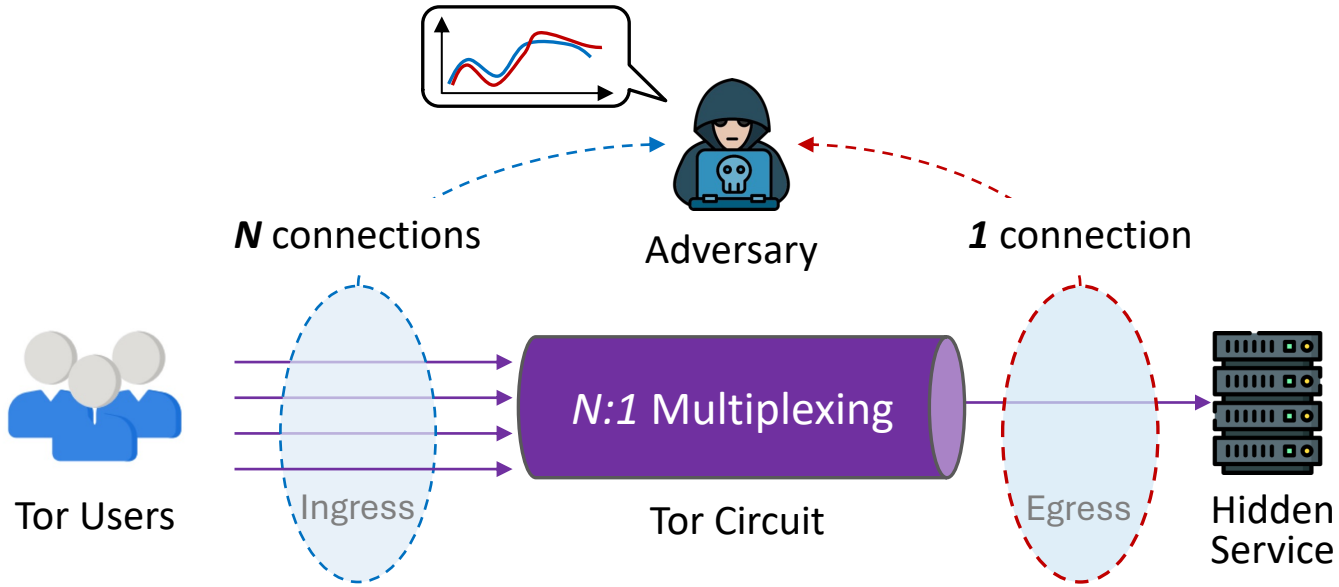
Bandwidth Overhead



Latency Overhead

Limitation: Lack of Dynamic Obfuscation for Egress Segments

- Existing obfuscation systems focus solely on ingress points
- Even with hidden services, Tor's static ***N:1*** multiplexing is weak to recent FCA (e.g., SUMO [NDSS '24])
- They ignore dynamic obfuscation at egress points that decrypt Tor traffic



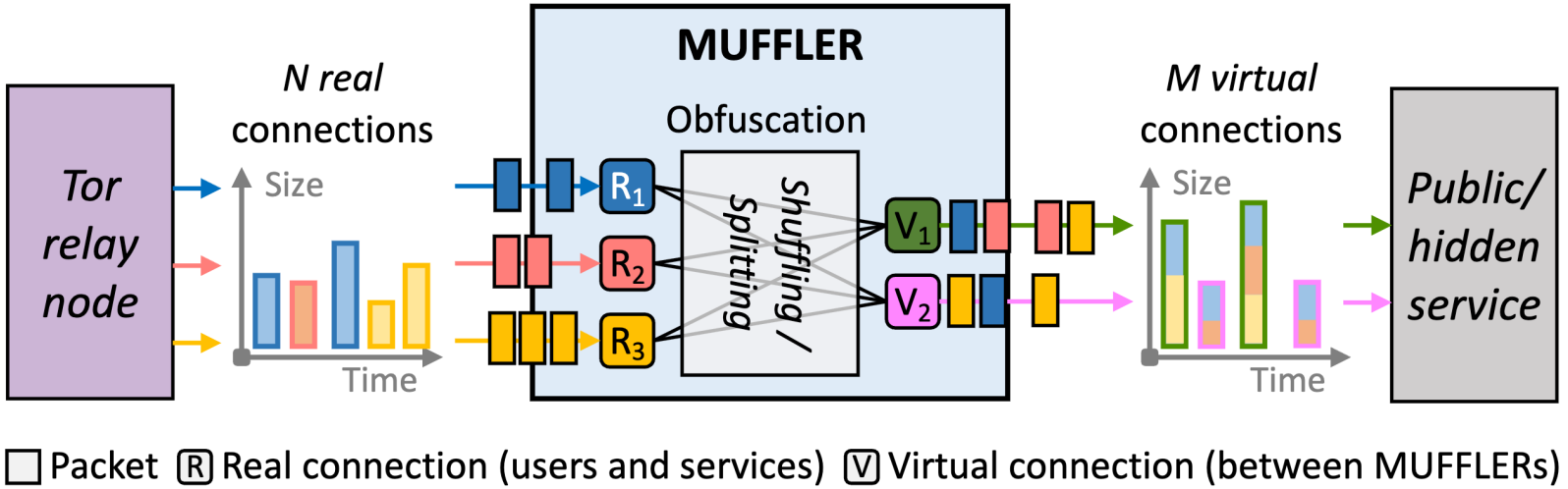
Design Considerations

- **Network efficiency:**
  - Obfuscate traffic without introducing intentional overhead
- **Egress robustness:**
  - Provide resilient and adaptive obfuscation at the egress point
- **Tor compatibility:**
  - Support seamless integration without modification of Tor binary or its protocols

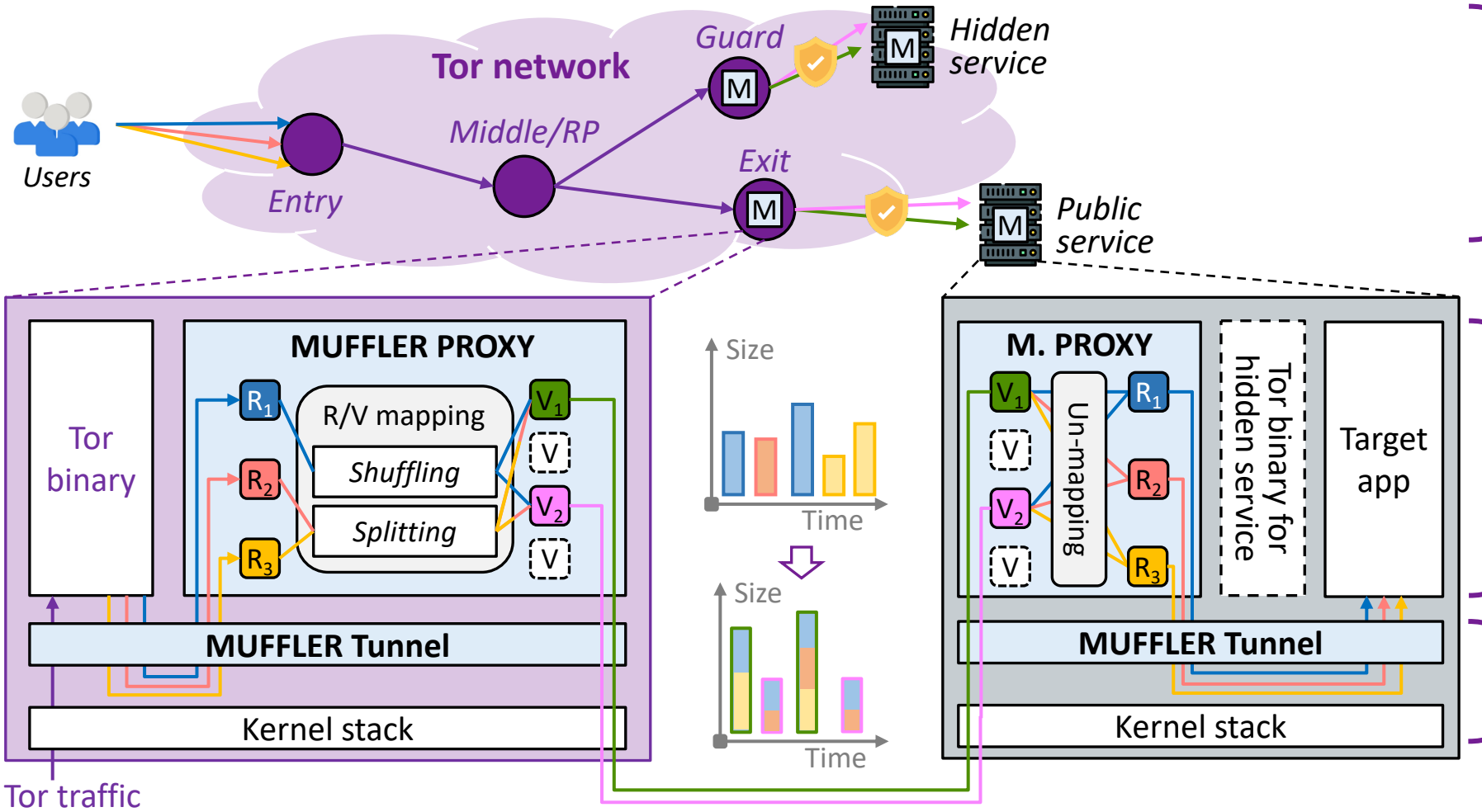
MUFFLER Overview

- A novel egress traffic obfuscation system tailored to Tor network
  - Maps ***N*** real connections to ***M*** virtual connections
  - Obfuscates traffic through connection **shuffling** and **splitting**

→ Attackers can only observe flows where packets from multiple clients are interleaved without adding intentional padding bytes or delays



MUFFLER Design



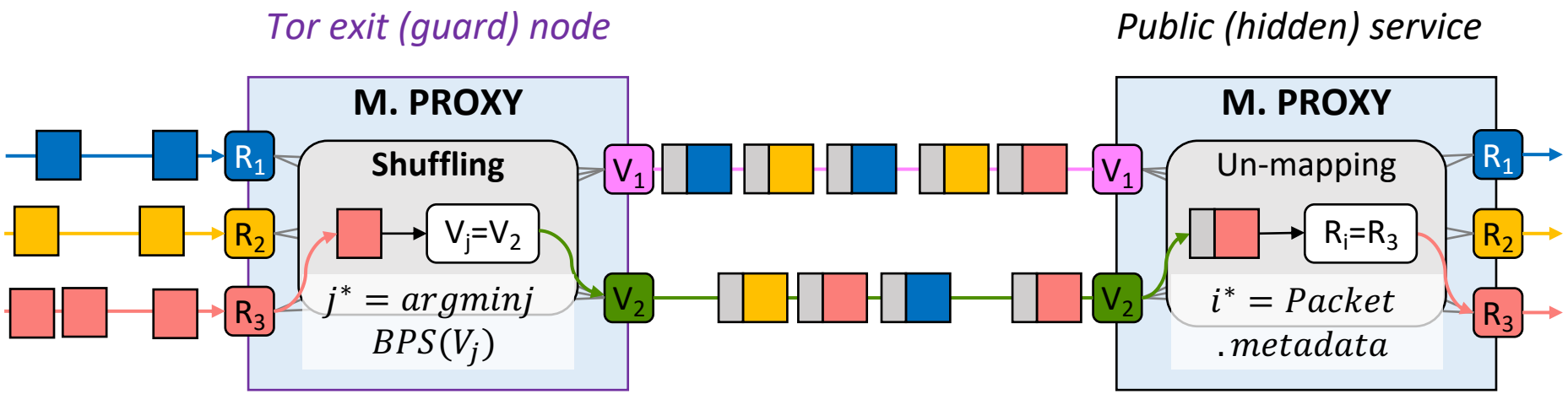
Deployed between the *last Tor node* and *the final server*

Connection-level obfuscation without intentional delays and paddings

Optimized critical path between the M. proxy and the Tor binary

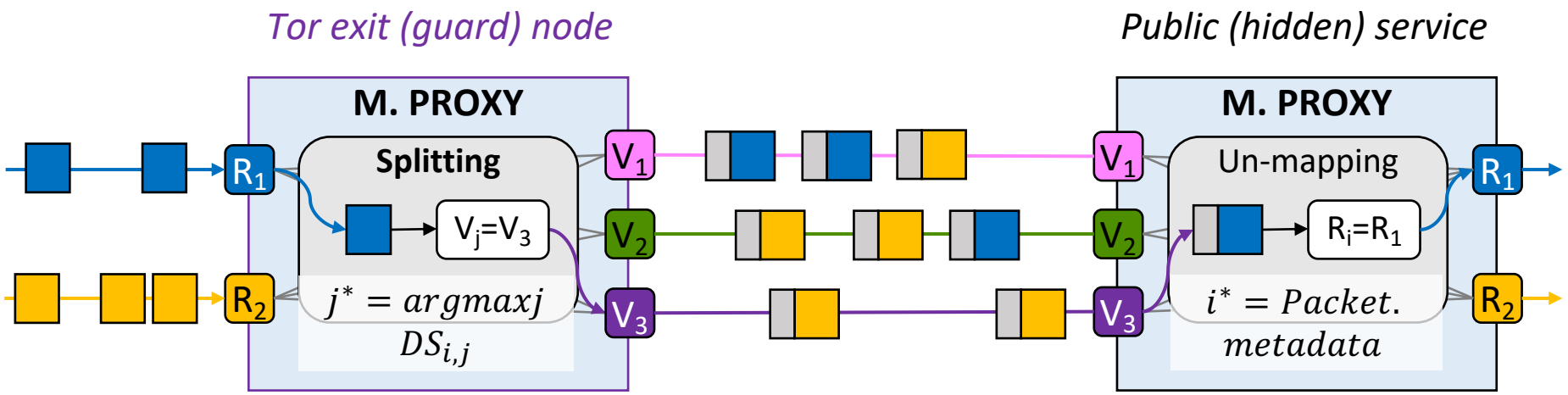
Connection-level Obfuscation

- **Shuffling:** *N* multiplexing real connections into *M* virtual connections (*N* > *M*)
  - A VC (*j*) is selected by considering current bandwidth of each VC
  - Maintaining similar BPS for each VC



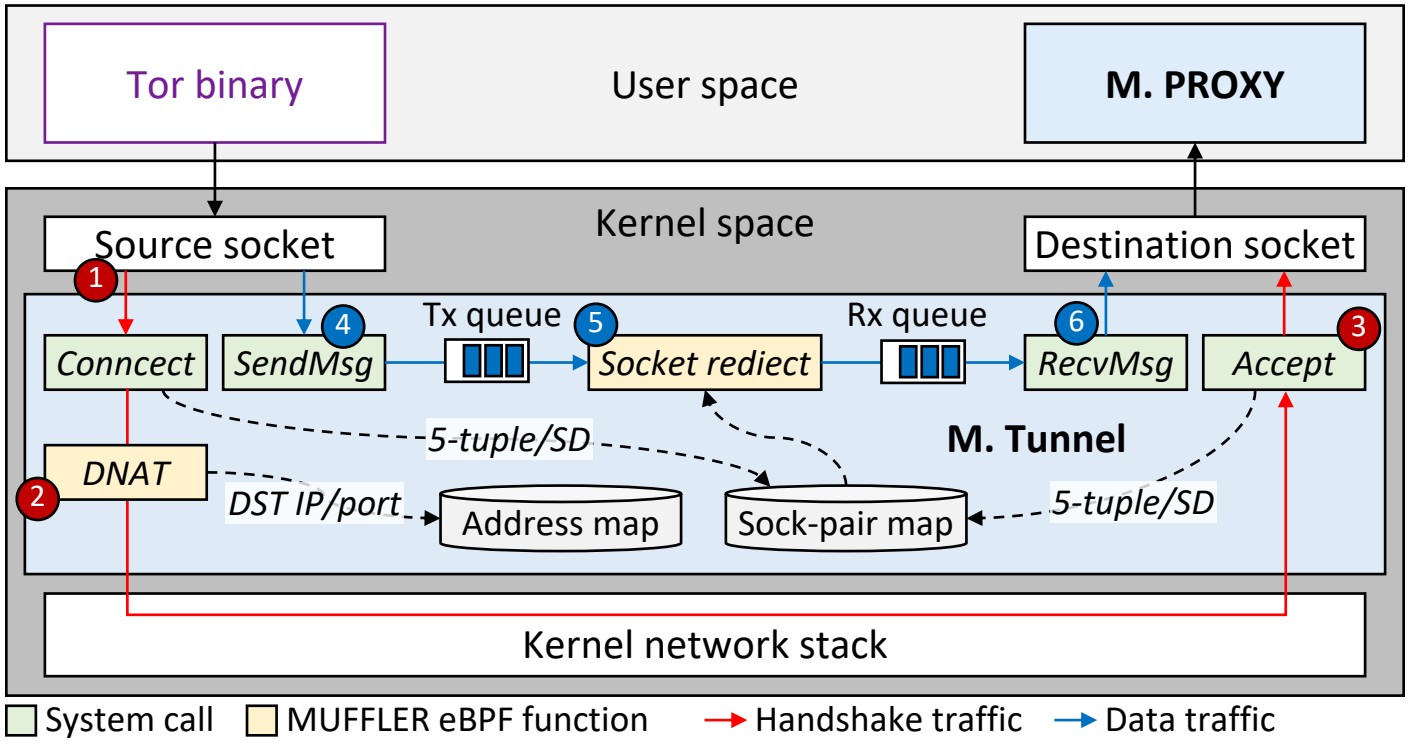
Connection-level Obfuscation

- **Splitting:** *N* multiplexing real connections into *M* virtual connections (*N* < *M*)
  - A VC (*j*) is selected by a bandwidth dissimilarity function
  - Maintaining spare packet density between virtual connections



Critical Path Optimization: MUFFLER Tunnel

- **eBPF-based packet redirection engine**
  - *Handshake packets* are forwarded through the original path
  - *Data packets* are redirected at the socket layer, bypassing the kernel stack



Implementation

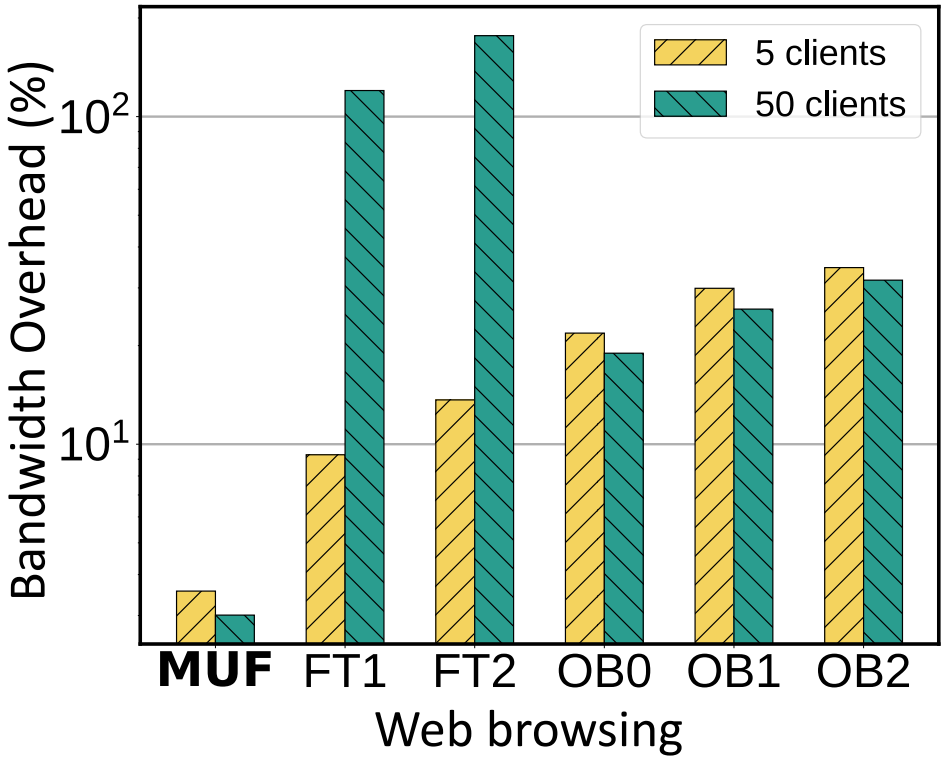
- **MUFFLER Proxy**
  - HAProxy
    - Extending it using the Go language
  - Control commands
    - *create, remove, relay,* and *keep-alive*
- **MUFFLER Tunnel:**
  - Leverages three eBPF programs
    - Traffic Control (TC), SOCK_OPS, and SK_MSG
- **Private Tor testbed:**
  - Tor relay nodes and directory authorities
    - Each machine is equipped with two Intel Xeon Silver 4114 CPUs with 64GB of memory
  - Public and hidden services
    - Each server is equipped with an Intel i9-10900X CPU with 256GB of memory

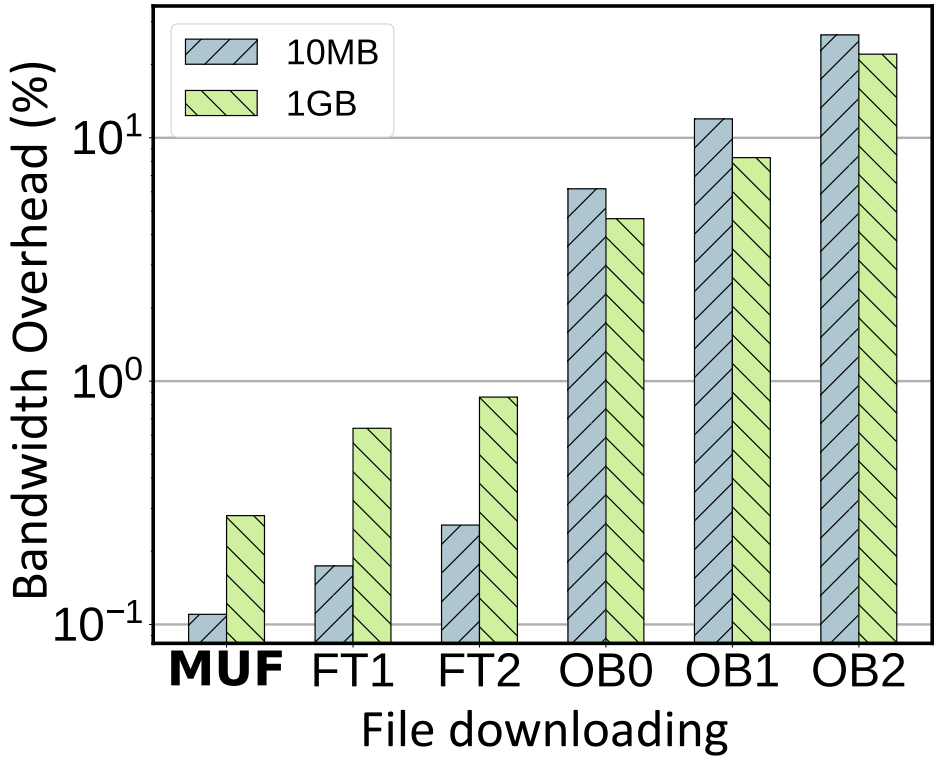
Defense Settings

	MUFFLER shuffling threshold	Obfs4 iat-mode	FRONT [USENIX SEC '20]
Defense settings	MUFFLER virtual connection = 4	0, 1, and 2	FRONT 1700 and FRONT 2500

Evaluation: Bandwidth Overhead

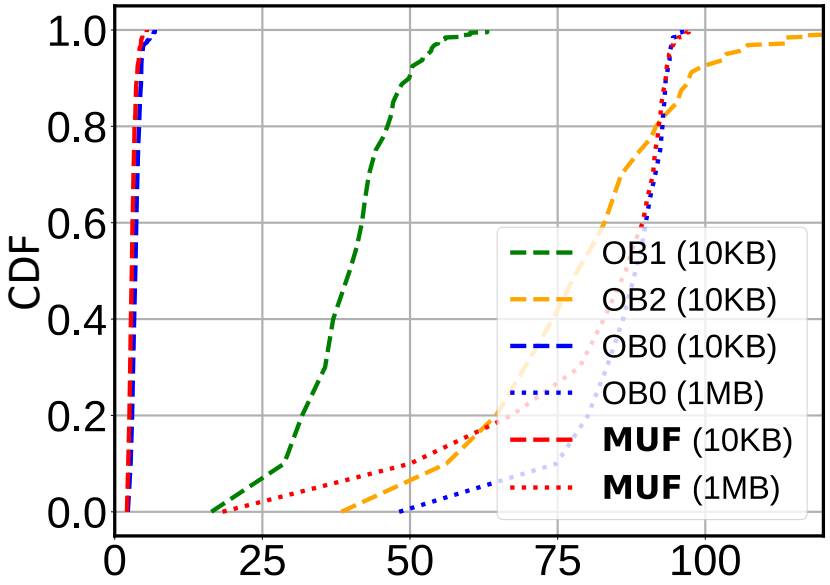
- **Website browsing:** Incurs around 3% bandwidth overhead, even with 50 clients
- **File downloading:** Maintains low overheads below 1%



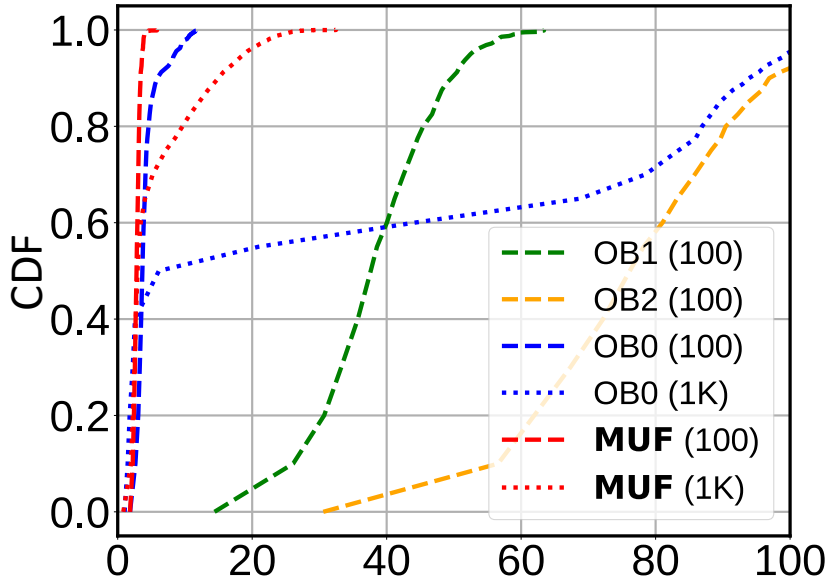


Evaluation: Latency Overhead

- **Latency (different file sizes):** 2.887ms mean latency for 10KB HTTP traffic
- **Latency (# of connections):** 3.4ms mean latency under 100 connections
- Both experiments show 27x faster performance than obfs4 iat-mode 2



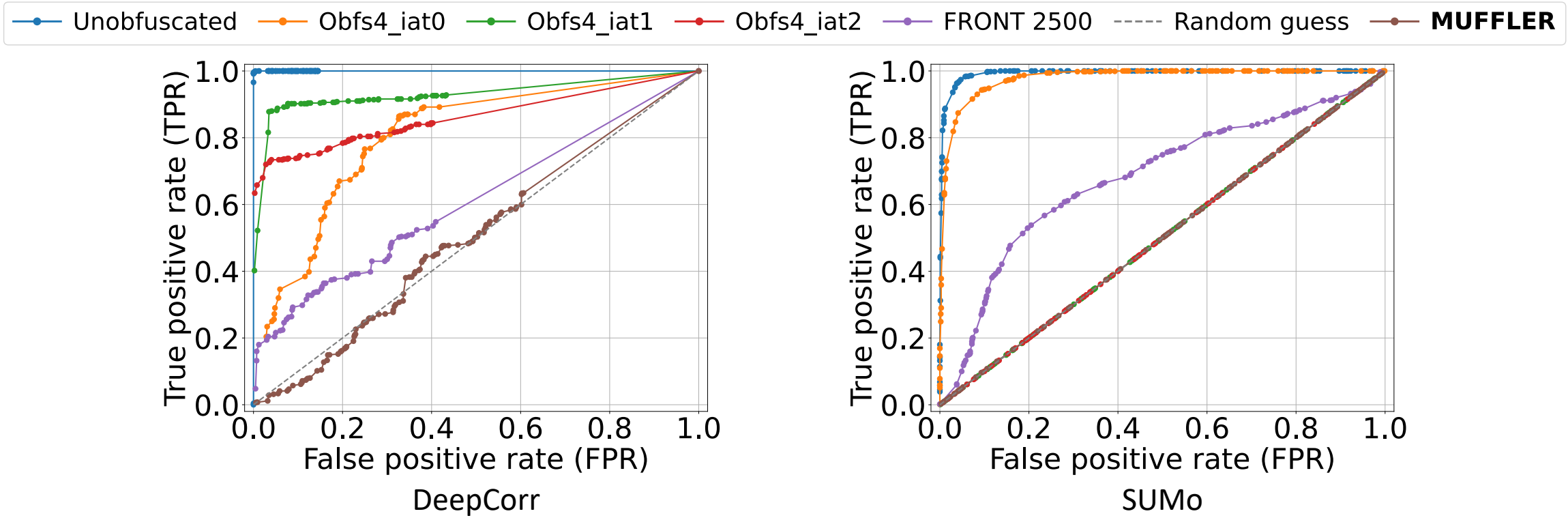
HTTP file sizes



# of concurrent connections

Evaluation: Obfuscation Effectiveness

- **Against the DeepCorr attack:** A TPR of 1% at an FPR of 10^{-2}
- **Against the SUMo attack:** A TPR of 1% at an FPR of 10^{-2}



Conclusion and Future Work

- **MUFFLER:**
  - Provides connection-level obfuscation through traffic shuffling and splitting at the egress point
  - Minimizes the critical path and eliminates intentional overhead without modifying the Tor binary
- **When there is a single real connection:**
  - The distribution of packets across virtual connections may reveal patterns
  - Fragmenting packets in the real connection into smaller segments
  - Embedding reconstruction metadata in segmented packets

Thank you for listening!

ms4060@etri.re.kr