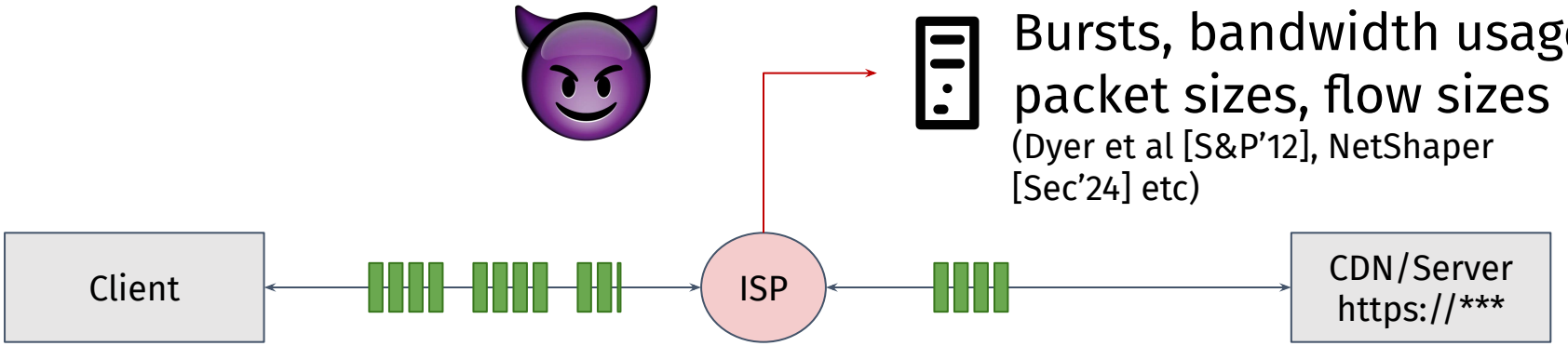
Rethinking the Role of Network Stacks for Website Fingerprinting Defenses

Elisaveta Lavrentieva, Marc Juarez, and Michio Honda
University of Edinburgh
ACM HotNets 2025, 18th November



What Website Fingerprinting does



Motivation

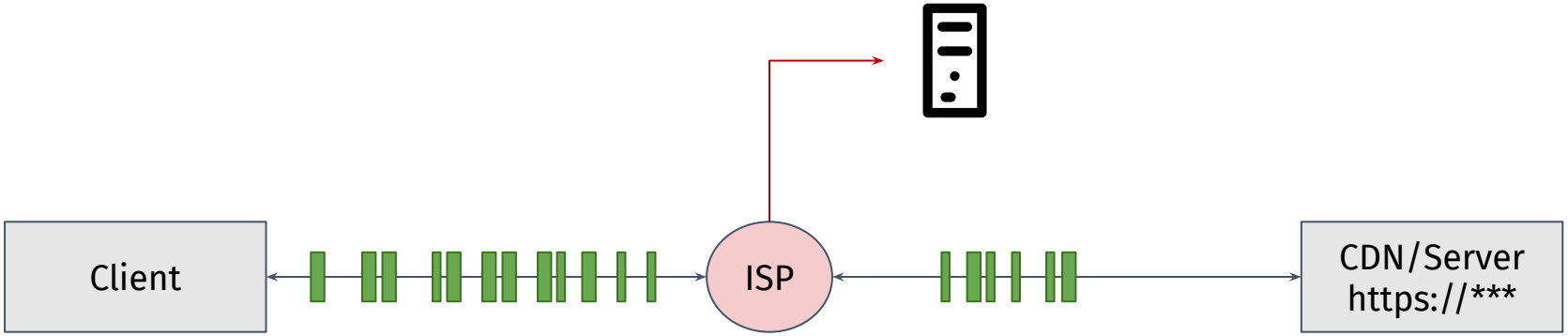
- Website fingerprinting (WF) over encrypted traffic is real
  - Inferring website/page identity the users visit

Motivation

- Website fingerprinting (WF) over encrypted traffic is real
  - Inferring website/page identity the users visit
    - National (e.g., censorship) and commercial (e.g., targeting ads) interest
    - Internet traffic is now more and more encrypted
      - TLS/QUIC, DoH, ECH

Protecting the users from WF seems crucial, but are the current defenses practical?

What the WF defenses want to do



Obfuscate traffic features

(Tamaraw [CCS'14], Surakav [S&P'22] etc)

WF defenses in reality

- Existing defenses operate at the app level

WF defenses in reality

- Existing defenses operate at the app level
  - **Inconsistent**
    - No guarantees that the stack generates intended packet sequences

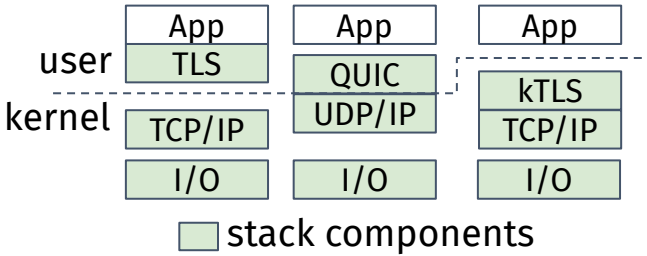
WF defenses in reality

- Existing defenses operate at the app level
  - **Inconsistent**
    - No guarantees that the stack generates intended packet sequences
  - **Inefficient**
    - App-limited flow must be enforced

App-level obfuscation is *inconsistent*

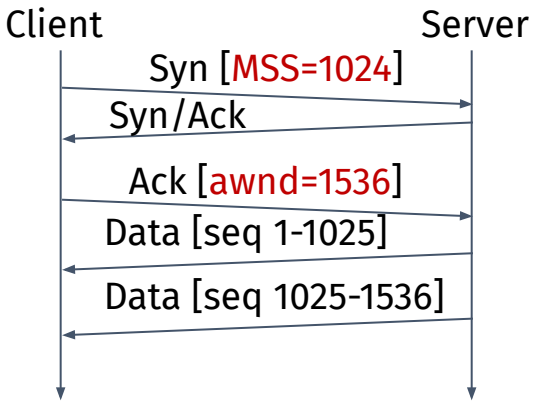
- Application data transmissions are asynchronous
  - Send buffering
    - Packetization based on PMTU
    - Transmissions ack-clocked
  - Segmentation Offload (TSO)
    - Micro bursts at a line rate
  - Packet scheduler
    - Fair queuing, pacing

True for various transport protocol organizations



App-level obfuscation is *inefficient*

- The application needs to enforce app-limited flows
  - Interleaved send operations
  - Small MSS
  - Small advertised window (awnd)
- HTTPOS [NDSS'11] example of enforcing 1024 and 512B packet burst:

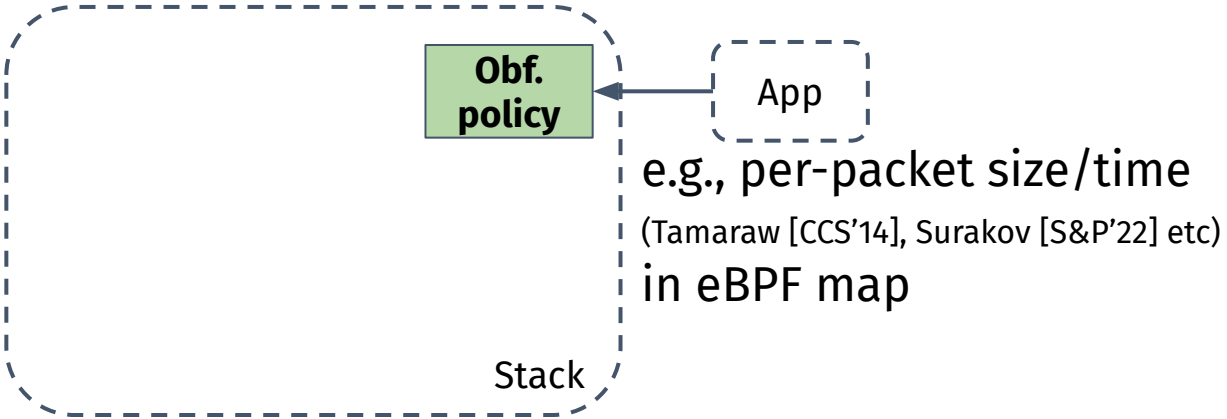


Stob: The case for **st**ack-level **ob**fuscation support

- New stack abstraction for packet sequence obfuscation

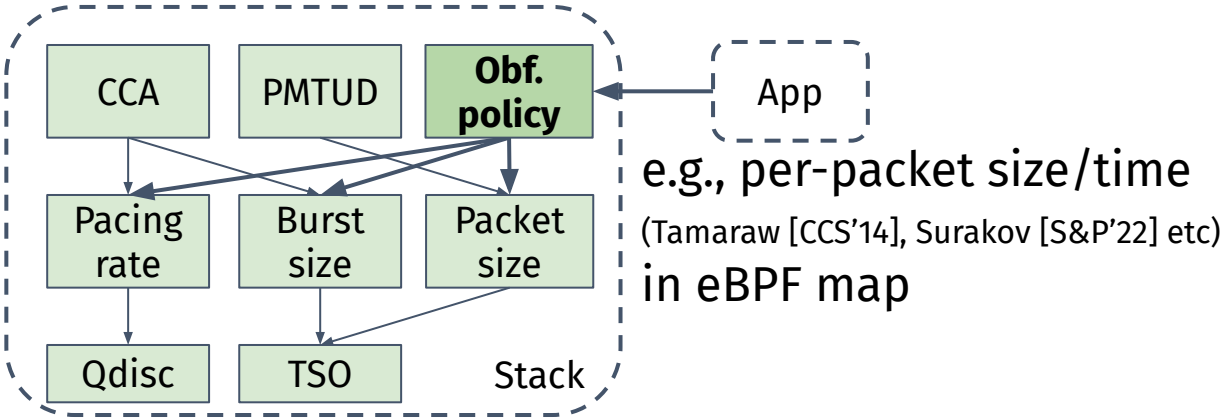
Stob: The case for **st**ack-level **ob**fuscation support

- New stack abstraction for packet sequence obfuscation
- App/admin creates and installs obfuscation policy



Stob: The case for **st**ack-level **ob**fuscation support

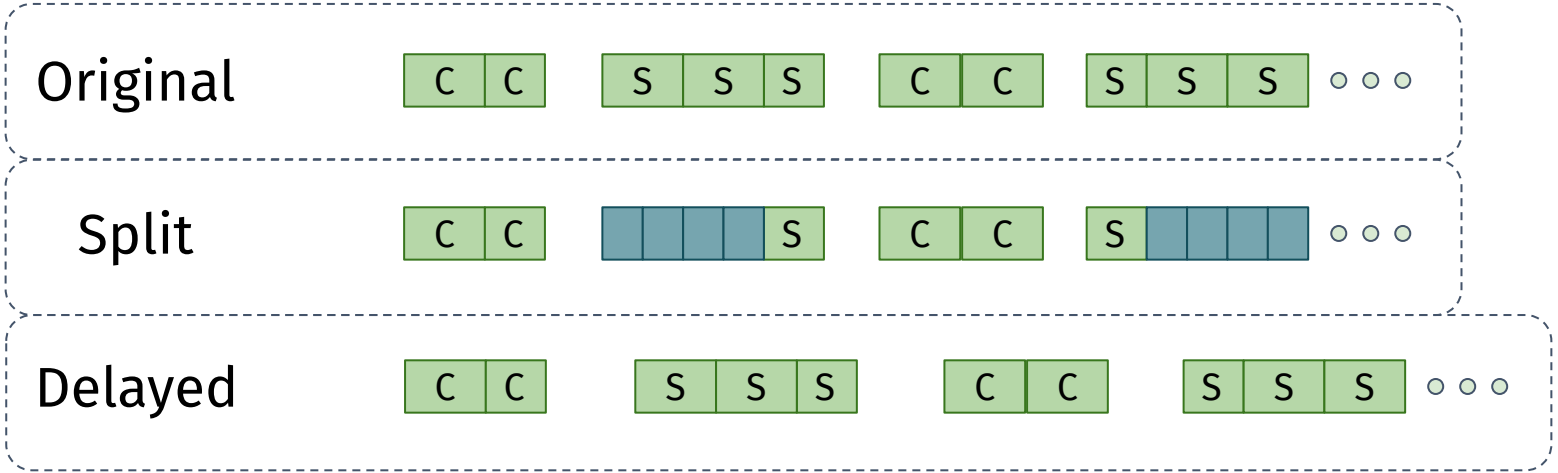
- New stack abstraction for packet sequence obfuscation
- App/admin creates and installs obfuscation policy
- Cooperate with other decisions [1]



[1] Making Linux TCP Fast, *Netdev conference 1.2*, 2016

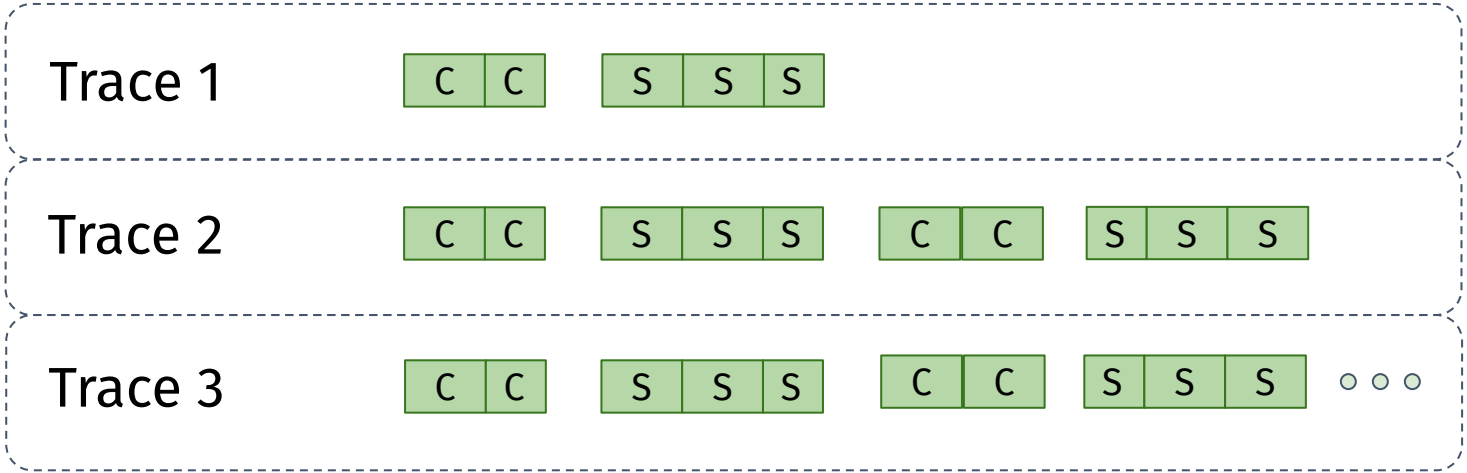
Preliminary experiment

- Simulate a server-side kernel website fingerprinting defense



Preliminary experiment

- Investigate the censorship scenario



Preliminary experiment

N	Original	Split	Delayed	Combined
15	0.798 ± 0.017	0.825 ± 0.024	0.825 ± 0.030	0.795 ± 0.031
30	0.884 ± 0.007	0.860 ± 0.013	0.855 ± 0.030	0.850 ± 0.062
45	0.938 ± 0.016	0.897 ± 0.030	0.913 ± 0.021	0.904 ± 0.004
All	0.963 ± 0.002	0.980 ± 0.008	0.980 ± 0.014	0.992 ± 0.009

Preliminary experiment

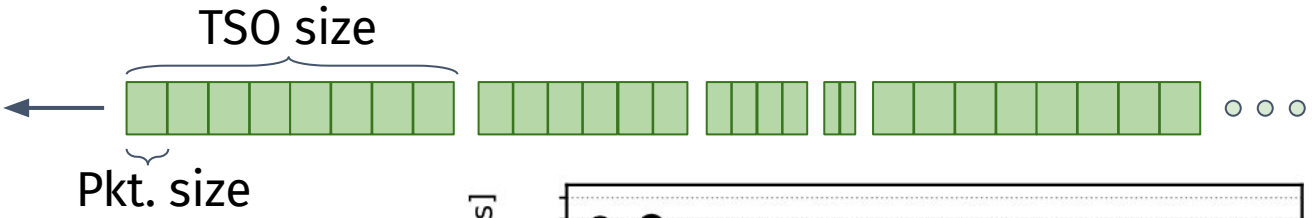
N	Original	Split	Delayed	Combined
15	0.798 ± 0.017	0.825 ± 0.024	0.825 ± 0.030	0.795 ± 0.031
30	0.884 ± 0.007	0.860 ± 0.013	0.855 ± 0.030	0.850 ± 0.062
45	0.938 ± 0.016	0.897 ± 0.030	0.913 ± 0.021	0.904 ± 0.004
All	0.963 ± 0.002	0.980 ± 0.008	0.980 ± 0.014	0.992 ± 0.009

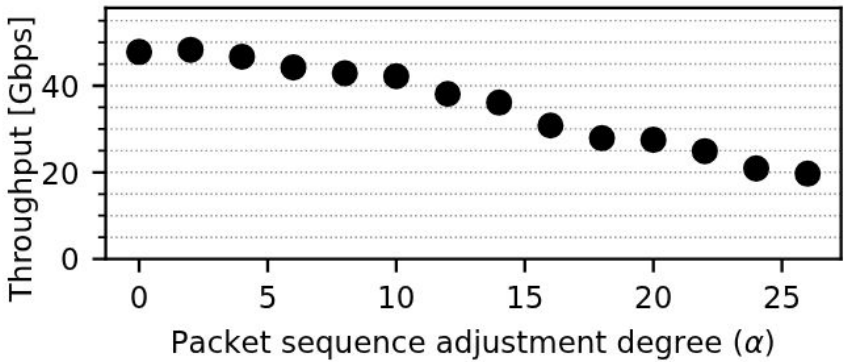
Performance implications

- Many defenses pre-generate target traces (e.g., per-packet size/time)
  - Tamaraw [CCS'14], Surakov [S&P'22] etc
- Transmission inefficiency is the main overhead

Performance impact of packet and TSO size

- Single flow (on a single core) over incremental reduction of TSO size (up to max(1, 44 - 2α)) and packet size (up to 1500 - 10α)





Sustain a high throughput range

Summary and research agenda

- Existing WF defenses are inconsistent or inefficient
  - Stack-level support for traffic obfuscation is needed

Summary and research agenda

- Existing WF defenses are inconsistent or inefficient
  - Stack-level support for traffic obfuscation is needed
- Deployment challenge
  - How to incentivise CDN operators?
    - Ensure low performance overheads
    - Guarantee differential privacy (NetShaper [Sec'24])

Summary and research agenda

- Existing WF defenses are inconsistent or inefficient
  - Stack-level support for traffic obfuscation is needed
- Deployment challenge
  - How to incentivise CDN operators?
    - Ensure low performance overheads
    - Guarantee differential privacy (NetShaper [Sec'24])
- CCA interplay challenge
  - How to avoid conflict or confusion with CCA's transmit decisions?
    - Make CCA obfuscation aware