Website Fingerprinting Defenses at the Application Layer

Giovanni Cherubin[1] Jamie Hayes[2] Marc Juarez[3]

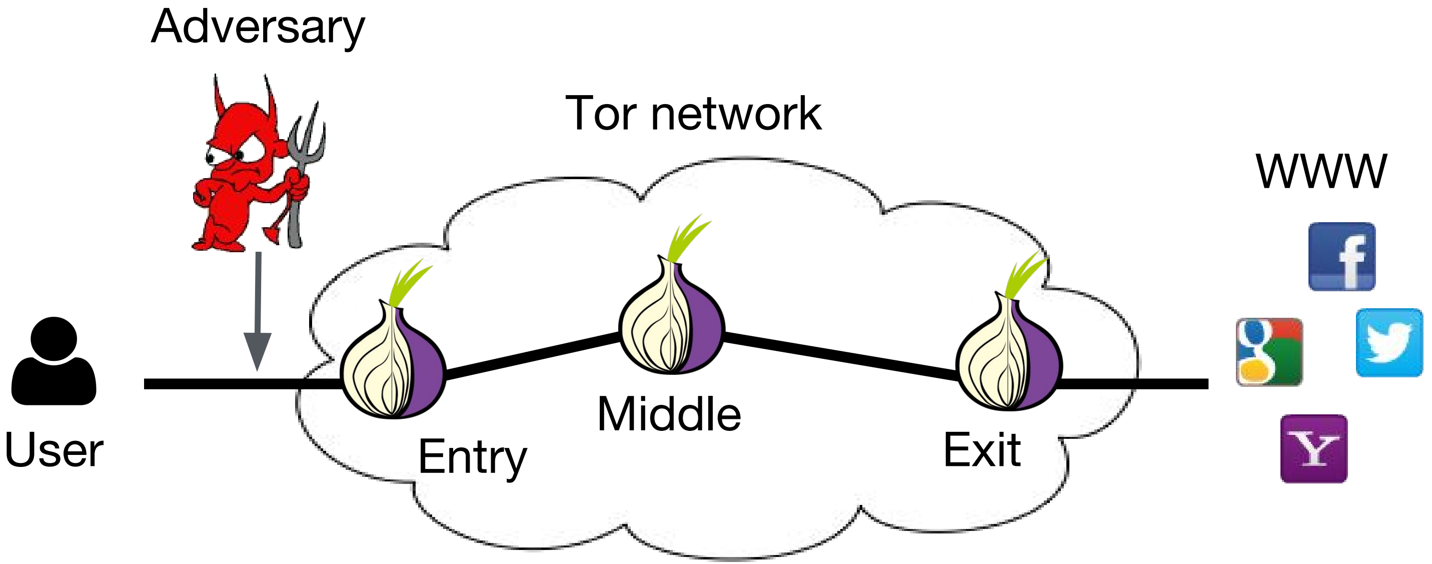
[1]Royal Holloway University of London

[2]University College London

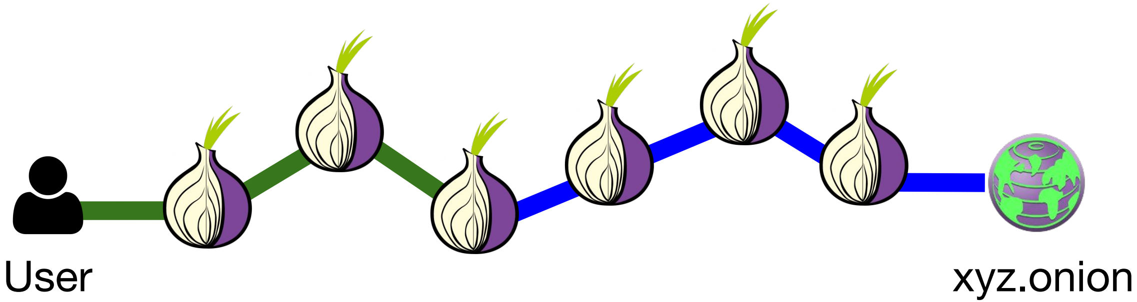
[3]imec-COSIC KU Leuven

19th July 2017, PETS'17, Minneapolis, MN, USA

Introduction: Website Fingerprinting (WF)



Tor Hidden Services (HS)



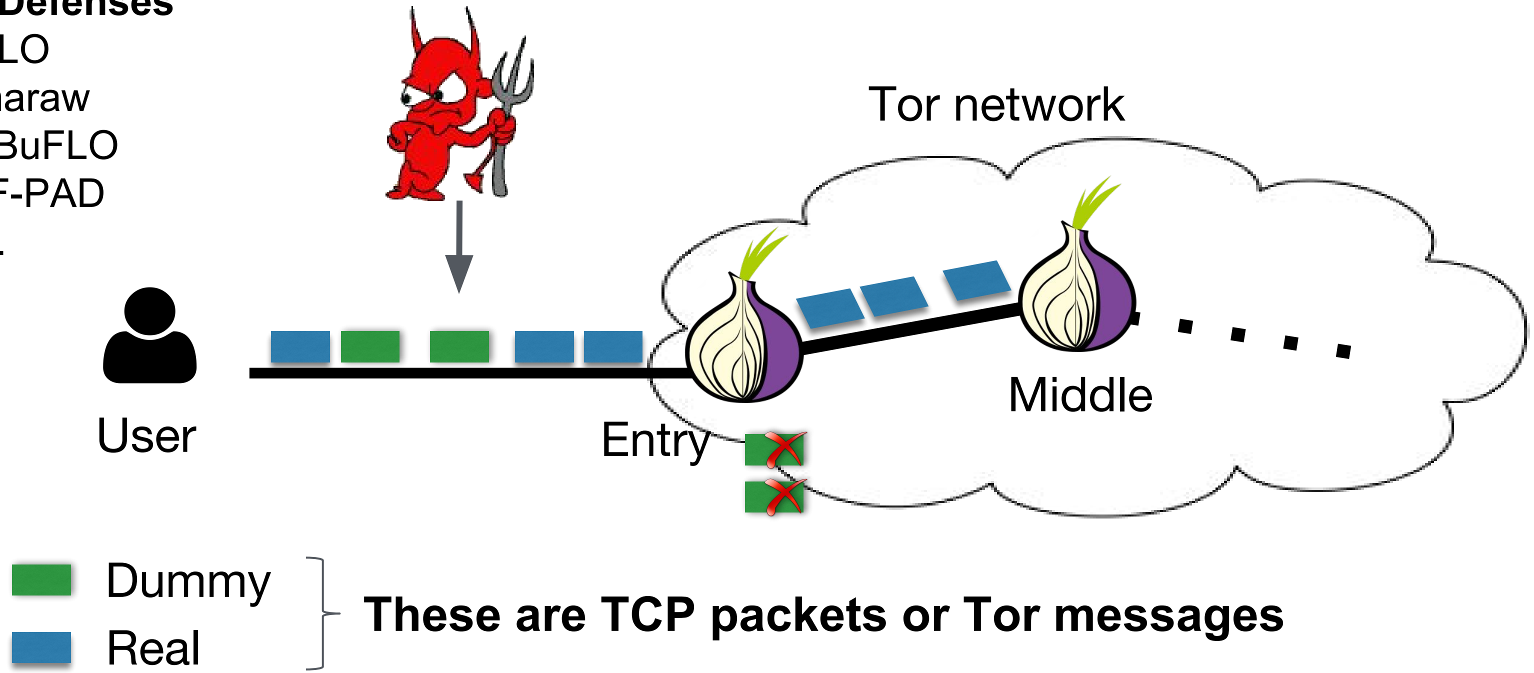
- HS: user visits xyz.onion without resolving it to an IP
- Examples: SecureDrop, Silkroad, DuckDuckGo, Facebook

Website Fingerprinting on Hidden Services (HSes)

- WF adversary can distinguish HSes from regular sites
- Website Fingerprinting in HSes is more threatening:
    - **Smaller world** makes HSes more identifiable
    - HS users vulnerable because content is **sensitive**

Website Fingerprinting defenses

WF Defenses
BuFLO
Tamaraw
CS-BuFLO
WTF-PAD
...

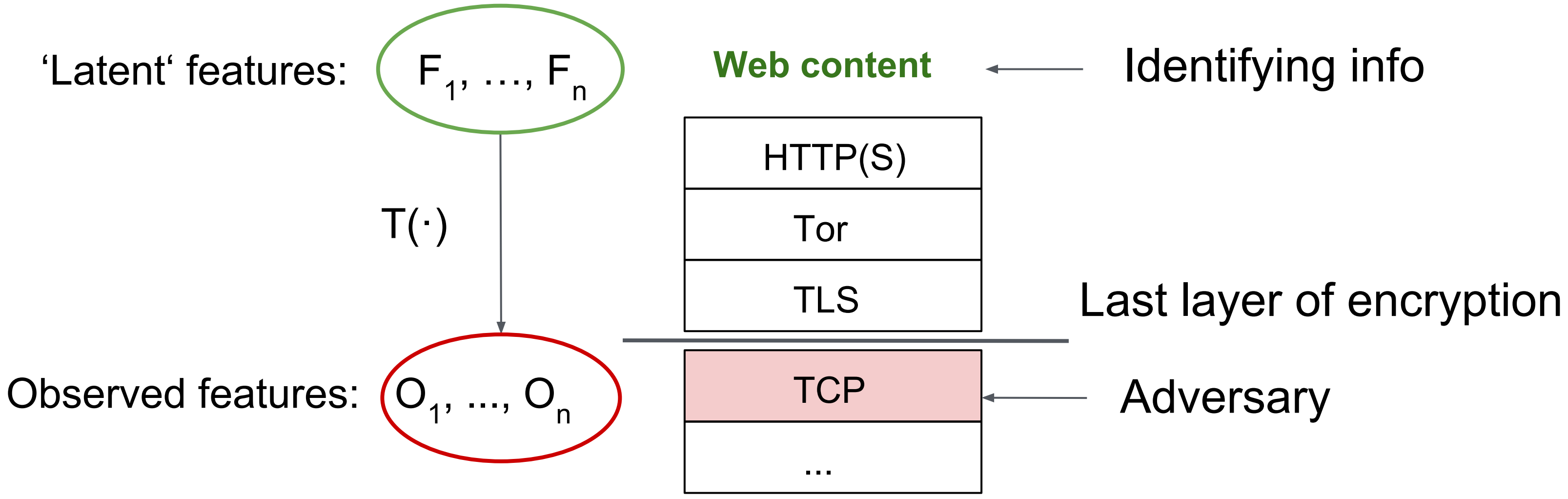


These are TCP packets or Tor messages

Application-layer Defenses

- Existing defenses are designed at the network layer

Key observation: identifying info originates at app layer!



Pros and Cons of app-layer Defenses

The main advantage is that they are easier to implement:

- do not depend on Tor to be implemented

Cons:

- padding runs end-to-end
- may require server collaboration:

...but HSes have incentives!

LLaMA

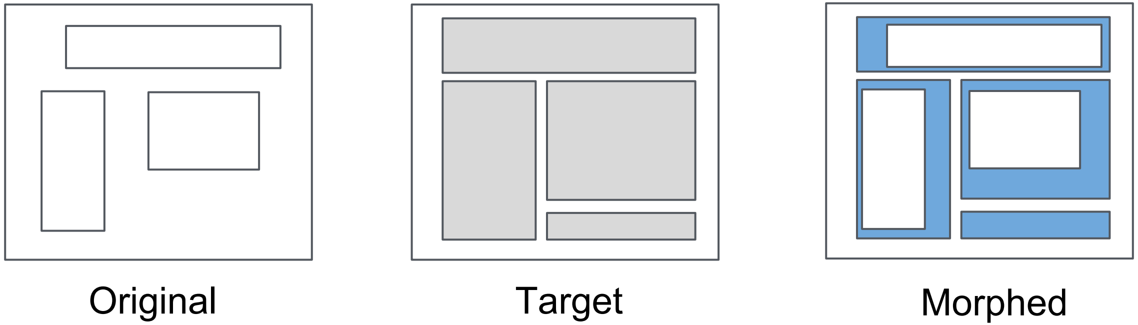
- **Client-side (FF add-on)**
- Applied on HTTP requests
- More **latency** overhead

ALPaCA

- **Server-side (first one)**
- Applied on hosted content
- More **bandwidth** overhead

(two different solutions, **not** a client-server solution)

ALPaCA



- Abstract web pages as **num objects** and **object sizes**:

  pad them to match a target page

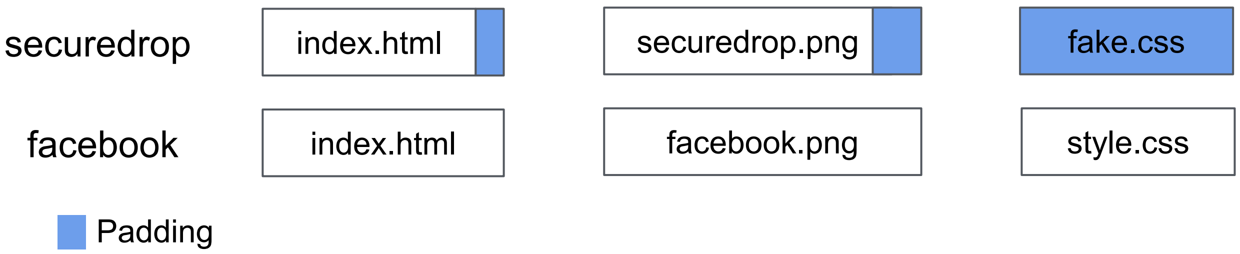
- Does not impact user experience:

  e.g., comments in HTML/JS, images' metadata, *hidden* styles

ALPaCA strategies (1)

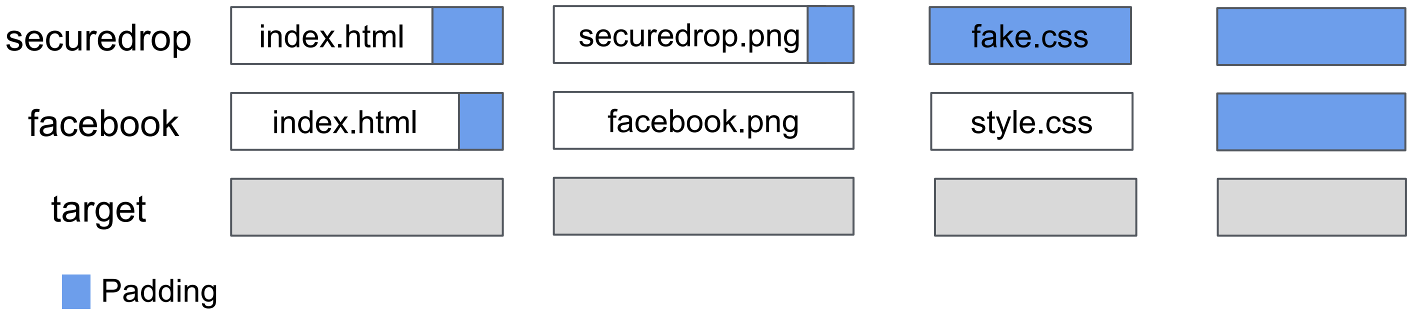
Example: protect a SecureDrop page

- Strategy 1: target page is Facebook



ALPaCA strategies (2)

- Strategy 2: pad to an “anonymity set” target page



Defines num objects and object sizes by:

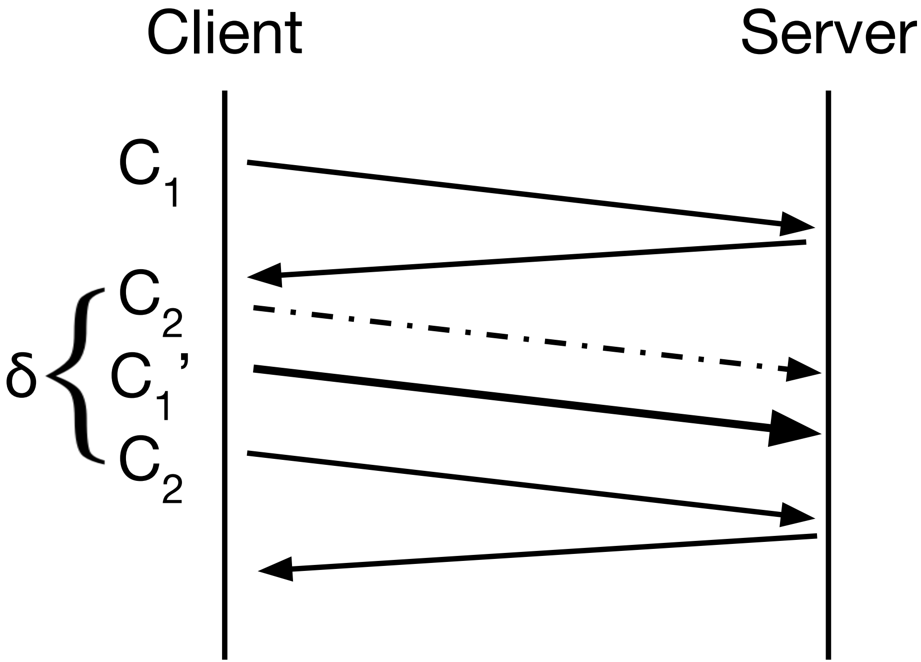
- Deterministic: next multiple of λ, δ
- Probabilistic: sampled from empirical distribution

LLaMA

- Inspired by Randomized Pipelining

  Goal: randomize HTTP requests

- Same goal from a FF add-on:
  - Random delays (δ)
  - Repeat previous requests (C_1)



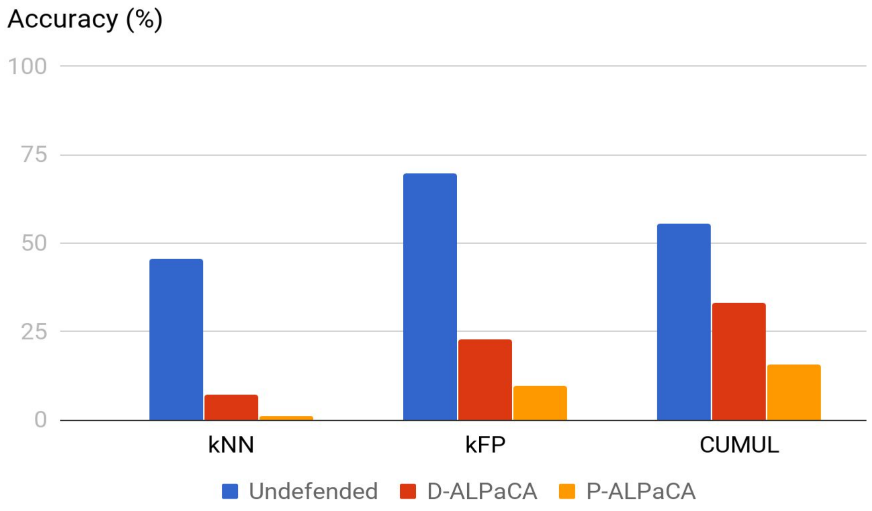
Evaluation: methodology

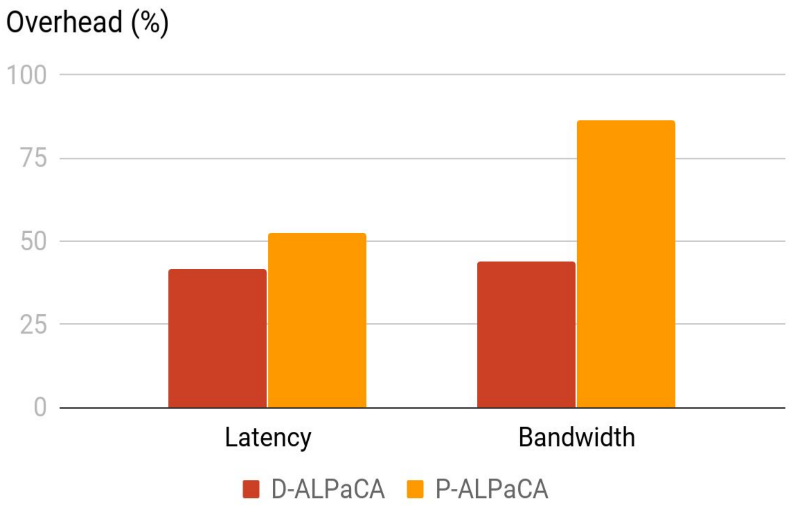
- Collect **with** and **without** defense: 100 HSes (cached)
    - Security: *accuracy* of attacks

      kNN, k-Fingerprinting (kFP), CUMUL
    - Performance: overheads
        - *latency* (extra delay)
        - *bandwidth* (extra padding/time)

ALPaCA: results

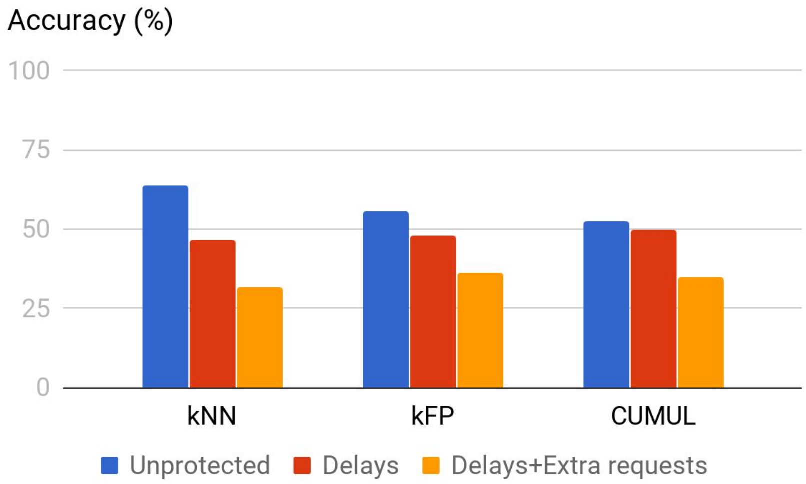
- From 60% to 40% decrease in accuracy
- 50% latency and 85% bandwidth overheads

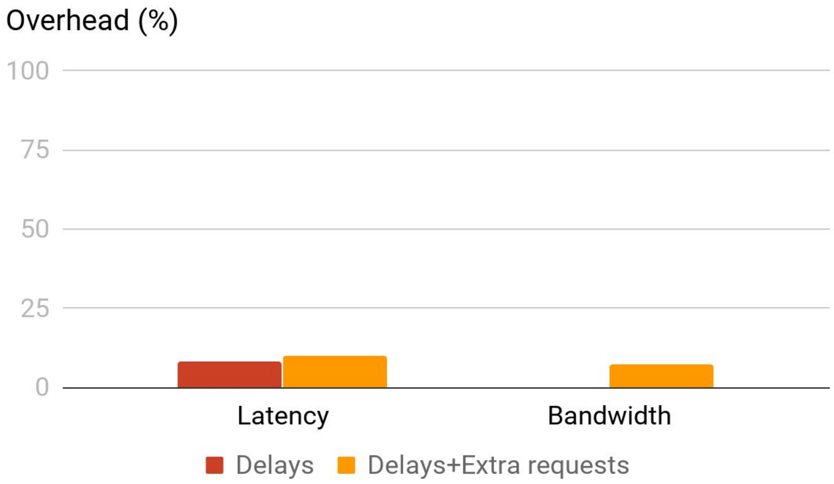




LLaMA: results

- Accuracy drops between 20% and 30%
- Less than 10% latency and bandwidth overheads





Take aways

- WF defenses at the app layer are **easier to implement**
- **HSes have incentives** to support server-side defenses:

    SecureDrop has implemented a prototype of ALPaCA

- ALPaCA is running on a HS: 3tmaadslguc72xc2.onion
- Source code: github.com/camelids