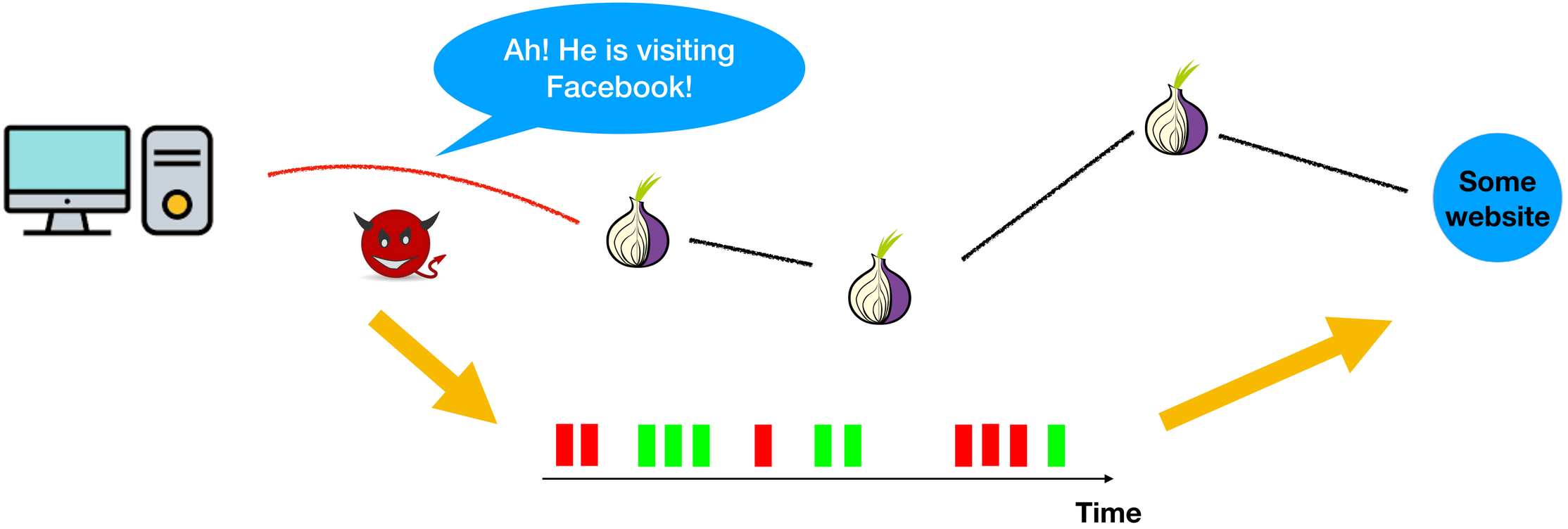
Surakav: Generating Realistic Traces for a Strong Website Fingerprinting Defense

Jiajun Gong[1], Wuqi Zhang[1], Charles Zhang[1], Tao Wang[2]
[1]The Hong Kong University of Science and Technology
[2]Simon Fraser University
{jgongac, wzhangcb, charlesz}@cse.ust.hk, taowang@sfu.ca

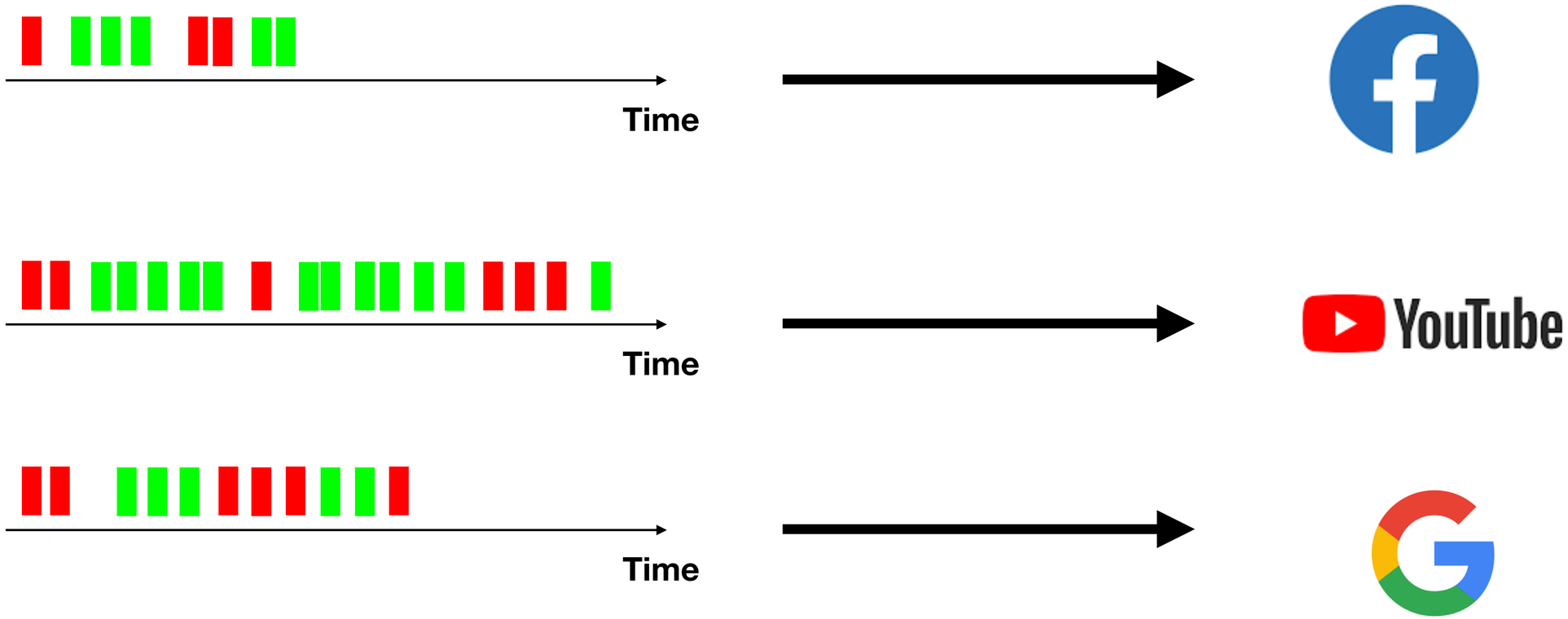
香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

SFU
SIMON FRASER UNIVERSITY

Website Fingerprinting



Website Fingerprinting (Classification Problem)



> 90% accuracy

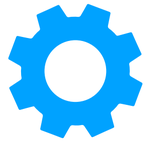
Existing Defenses

Defense	Overhead	Attack accuracy/ recall	Limitation
WTF-PAD	Low	80-90%	Weak protection
FRONT	Low	40-70%	
Walkie-Talkie	Low	At most 50%	Hard to implement
Tamaraw	High	~10%	

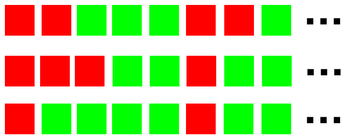
Limitation & Motivation

* Tamaraw: Fixed sending pattern → **Unaffordable overhead**

* Walkie-Talkie: Prior knowledge of pages → **Hard to implement**

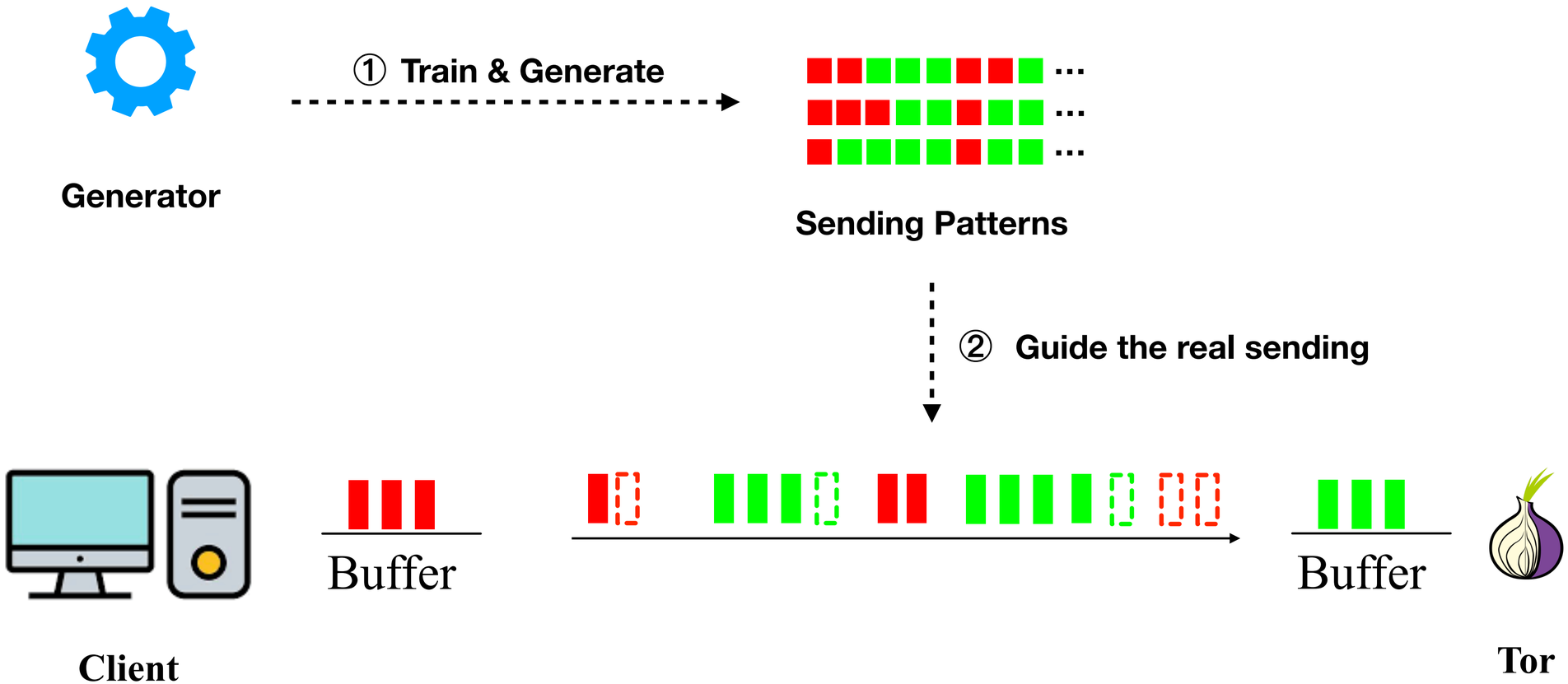


Generative model



Sending Patterns

Surakav: a new defense



Phase 1: Generator Training

Questions:

- **What pattern to mimic?**

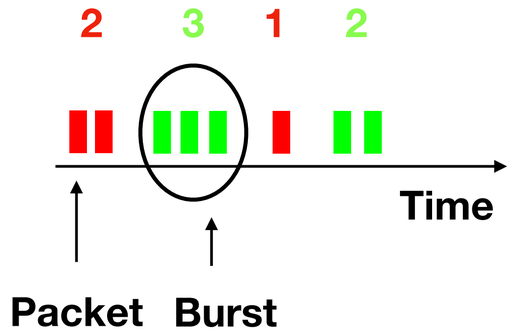
  Realistic burst sequence

  - **close to real loadings**
  - **Training data is easy to get**

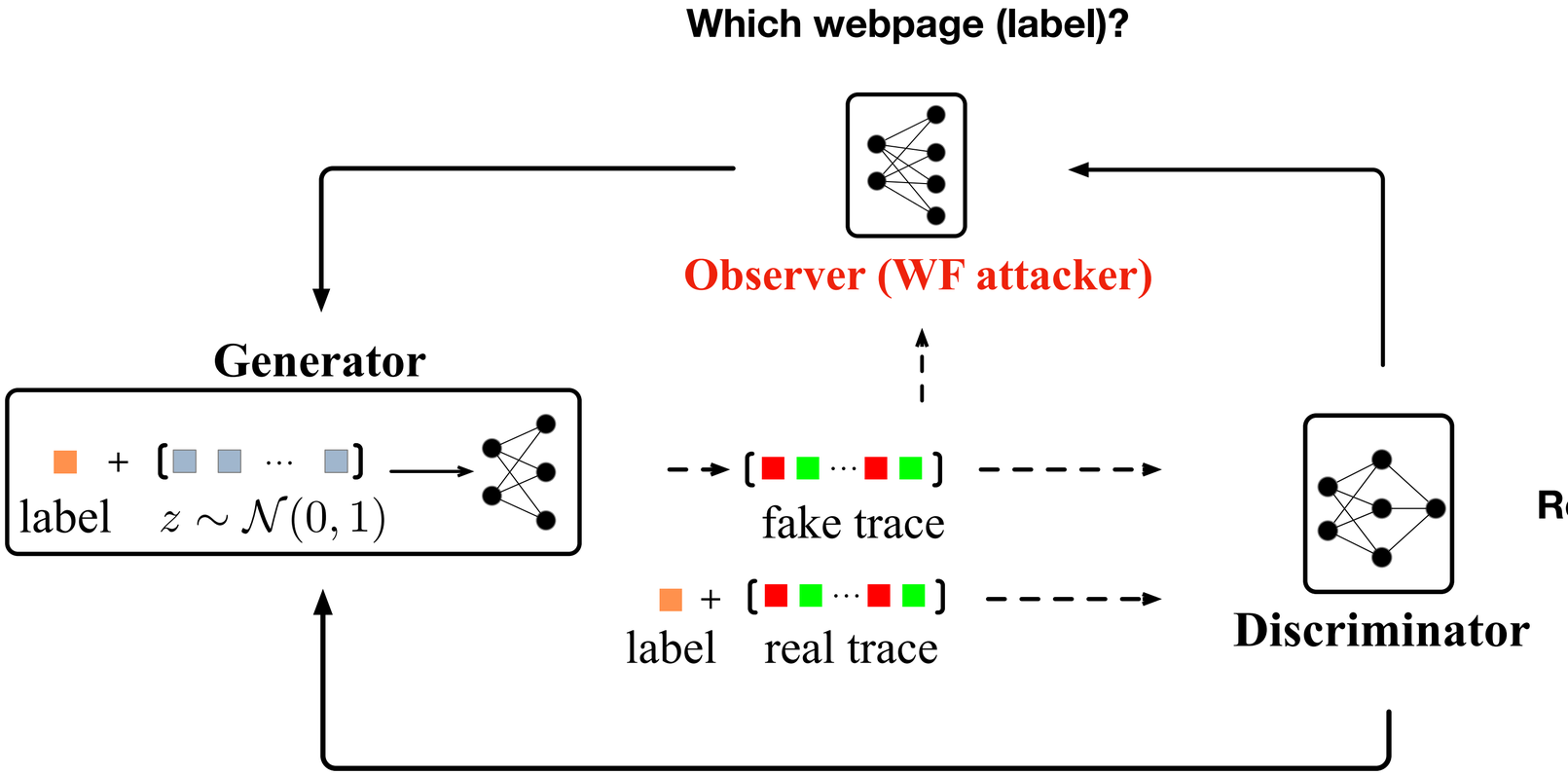
- **What generative model to use?**

  Generative Adversarial Network (GAN)

  - **More diverse**
  - **More realistic**



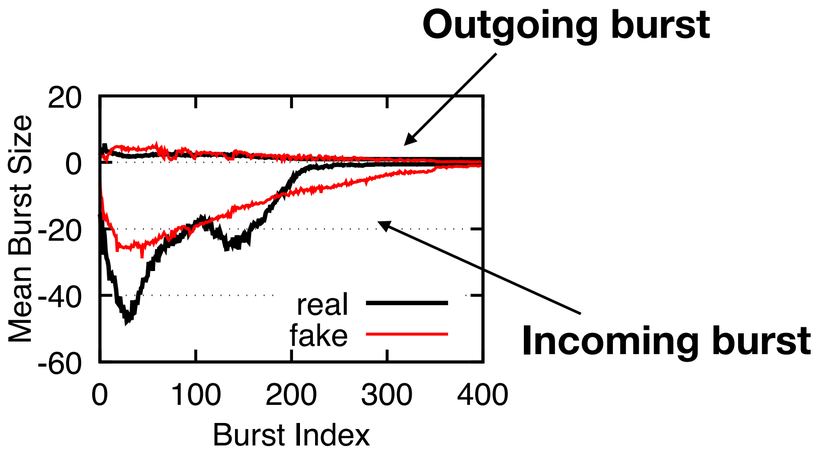
Phase 1: Generator Training



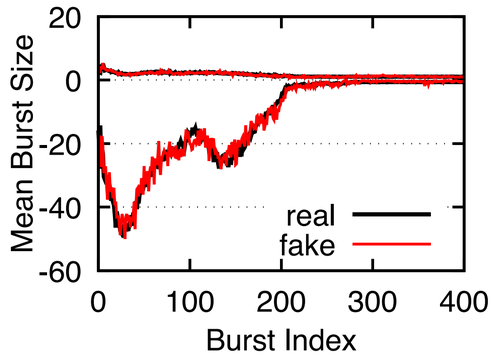
$$Loss_G = Loss_D - \alpha\, Loss_O$$

Generator Performance

- Rimmer's dataset (2017) 100 x 1000
- Wasserstein Distance 0.9 -> 0.02
- Generated traces can fool the observer at a 90% success rate



After Epoch 1

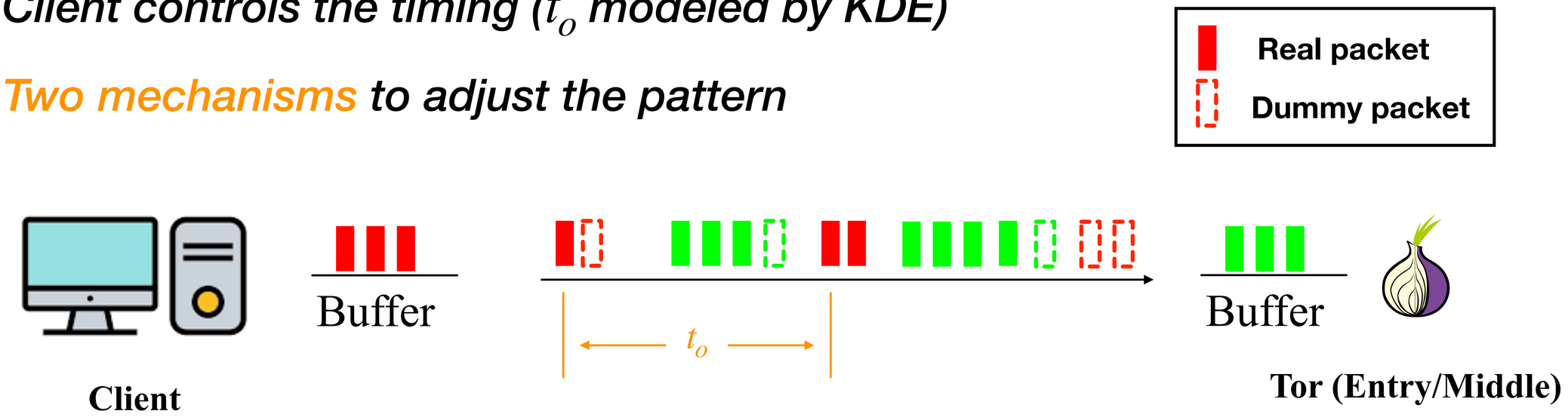


After Epoch 500

the fake traces are statistically close to the real ones.

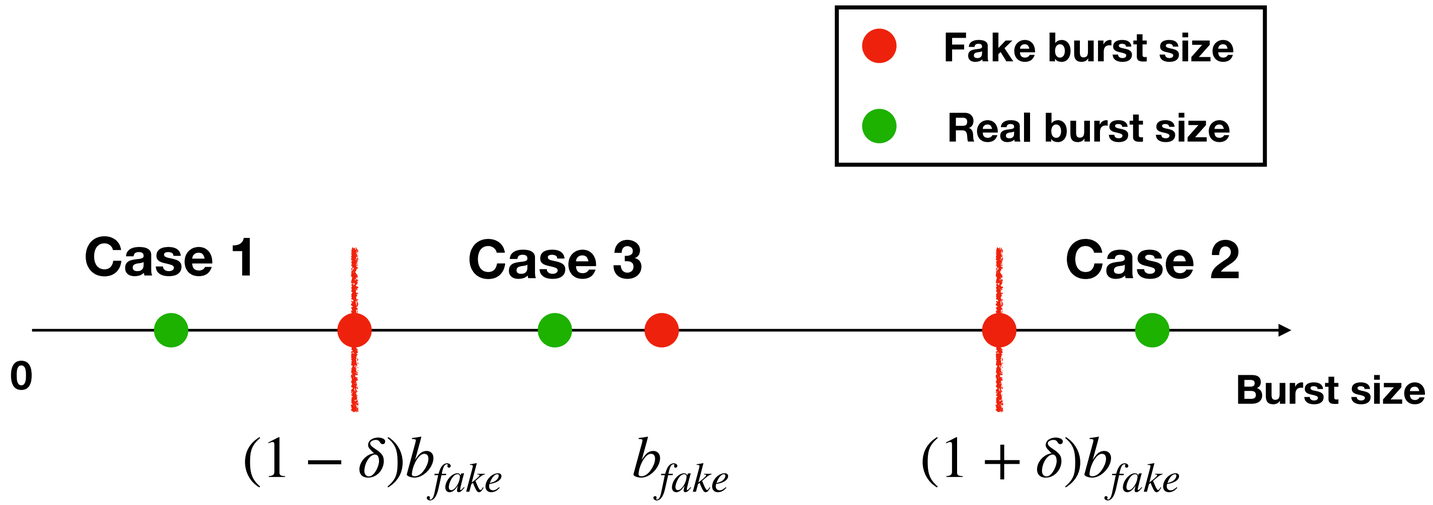
Phase 2: Packet Regulation

- Generate a trace *T* from Generator
- Send bursts of data based on *T*
  - *Client controls the timing (t_o modeled by KDE)*
  - *Two mechanisms to adjust the pattern*



Phase 2: Packet Regulation

- **Burst Size Adjustment (δ)**
  - Case 1: Send $(1-\delta)b_{fake}$
  - Case 2: Send $(1+\delta)b_{fake}$
  - Case 3: Send b_{real}

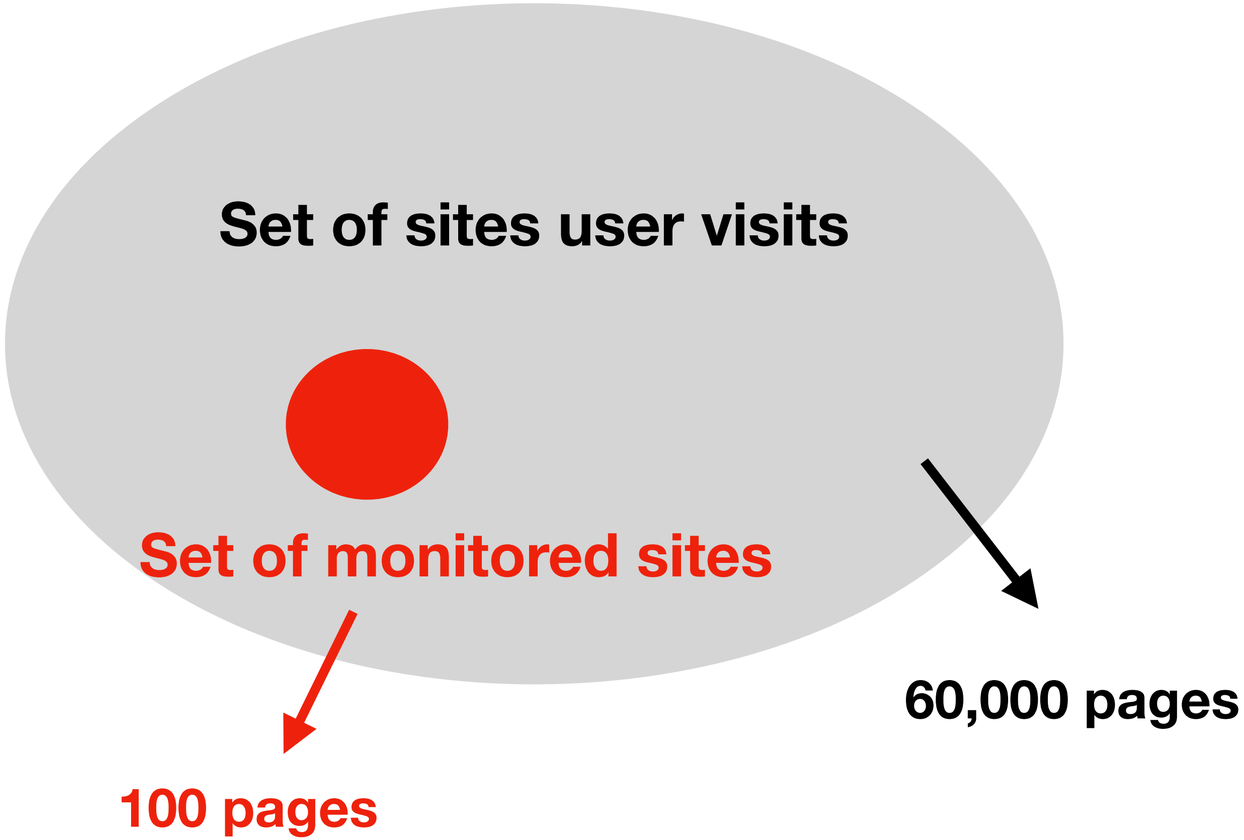


Phase 2: Packet Regulation

- **Random Response (Proxy side)**
  - **Triggered when buffered real data** $b_{real} = 0$
  - 50% chance to skip sending the dummy burst
  - 50% chance to send $(1 - \delta)b_{fake}$

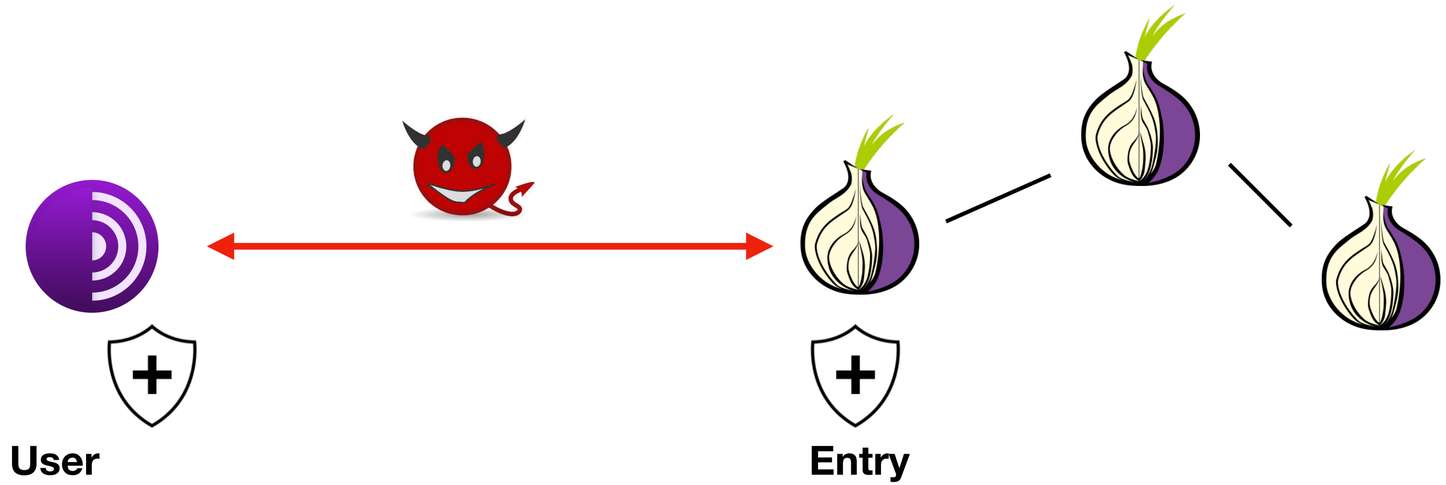
Experiment Setup

- Open-world setting
- Crawled from Tranco list



Experiment Setup

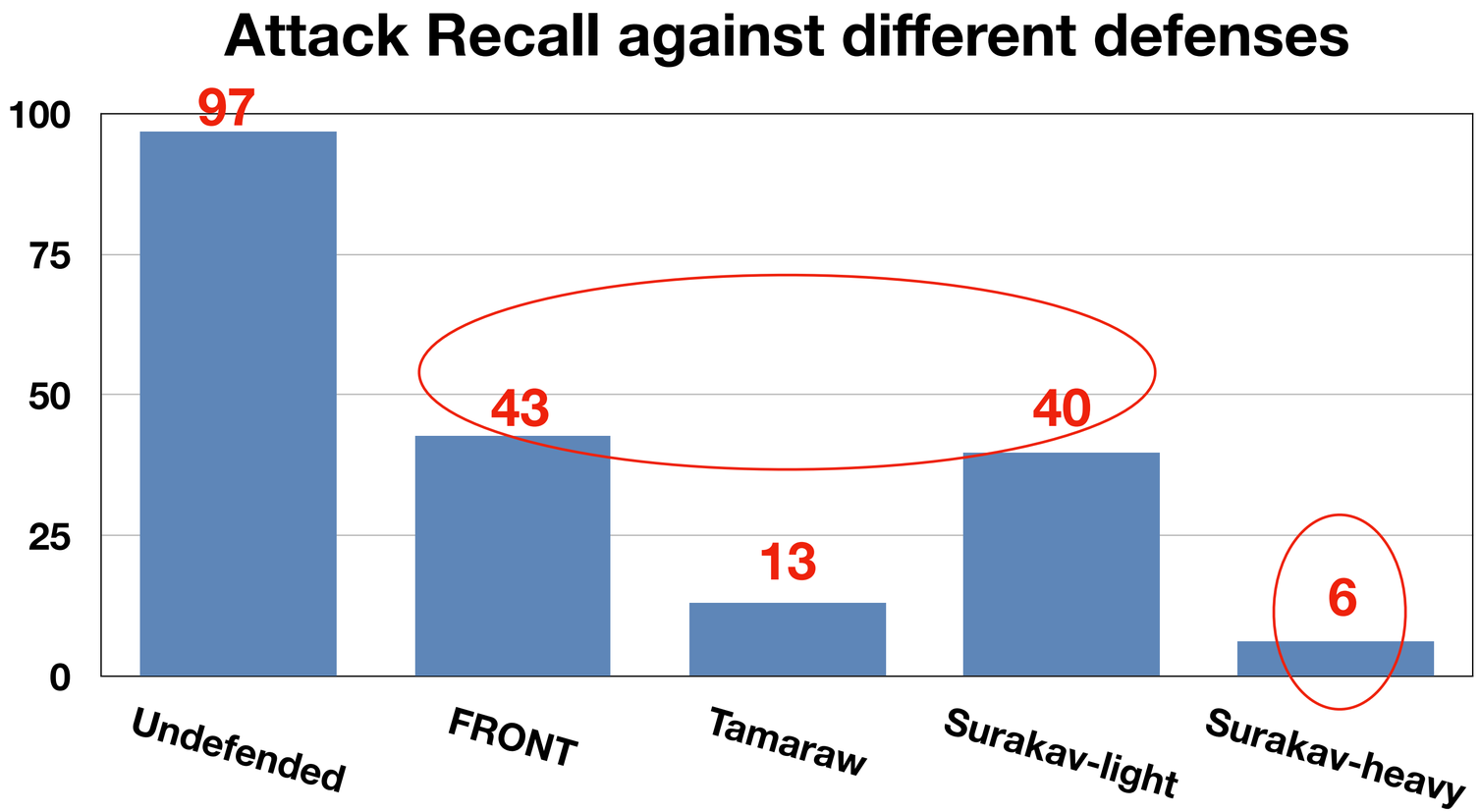
- Test in the real Tor network
- Each defense is implemented as Pluggable Transport
- Client in Hong Kong, Entry in the US.



Surakav Performance against different attacks

	Surakav-light ($\delta = 0.6$)		Surakav-heavy ($\delta = 0.4$)	
	TPR (%)	**FPR (%)**	**TPR (%)**	**FPR (%)**
kFP	0.85	0.02	0.01	0
CUMUL	11	9	3	8
DF	39	6	8	3
Tik-Tok	40	4	6	1

Surakav Performance comparing to other defenses



Defense Performance

Defense	Data Overhead	Time Overhead
FRONT	97	0
Tamaraw	121	26
Surakav-light	55	16
Surakav-heavy	81	17

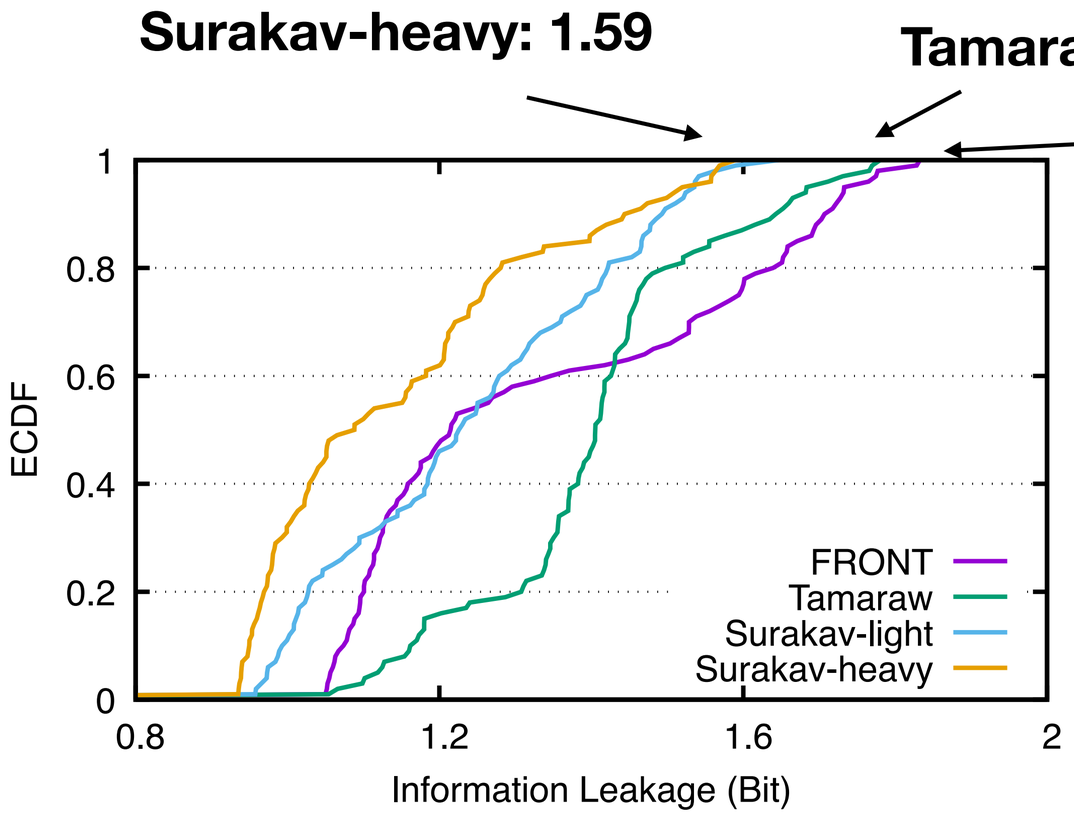
Compared to FRONT:

- **42% less data overhead and a similar protection rate (43% -> 40% TPR)**
- **Similar overhead offers more robust protection (43% -> 6% TPR)**

Compared to Tamaraw:

- **40% less data overhead and 10% less time overhead. (13% -> 6% TPR)**

Information Leakage Analysis

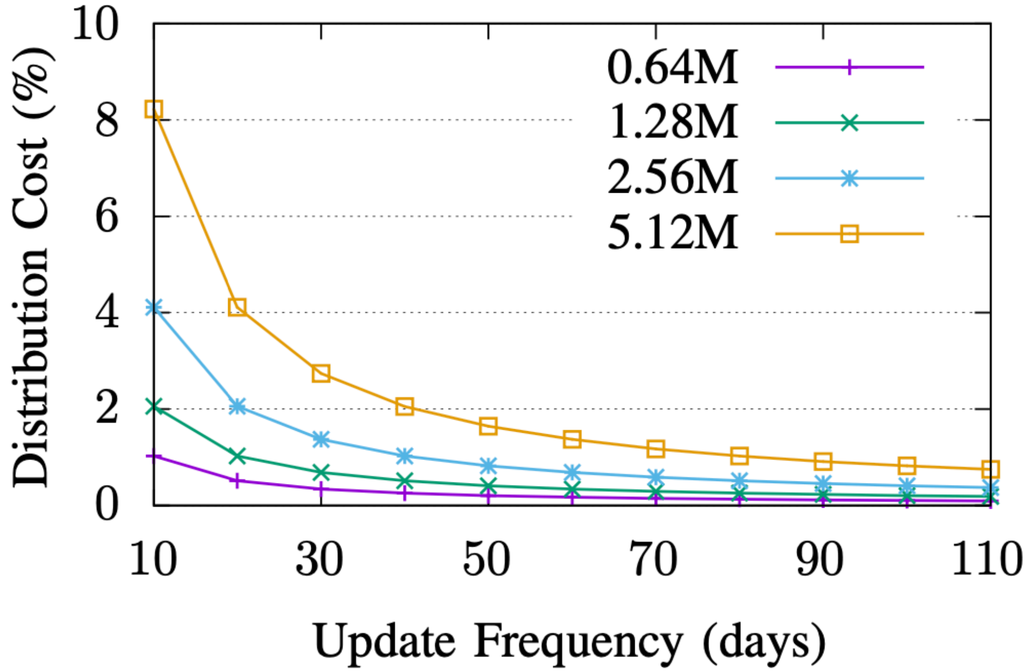


- examines the amount of information learnt from a specific feature
- WeFDE framework (Li et al., CCS' 18)

Surakav leaks the least bits of information for the top 100 informative features!

Distribution Cost

- The trained model is ~ 3 MB.
- Suppose the model is distributed by the Tor directory servers



1~8 % bandwidth overhead

Summary

- **Propose a strong WF defense Surakav**
  - Leverage a self-designed Generative Adversarial Model
  - Two random mechanisms to dynamically adjust the sending patterns

Code is available at

- GAN training: https://github.com/websitefingerprinting/wfd-gan
- Implementation: https://github.com/websitefingerprinting/surakav-imp