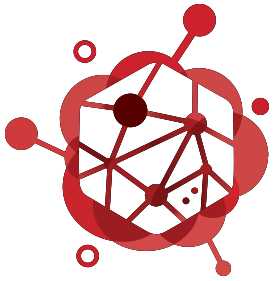


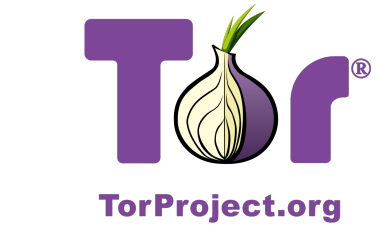
SAND Lab

sandlab.cs.uchicago.edu

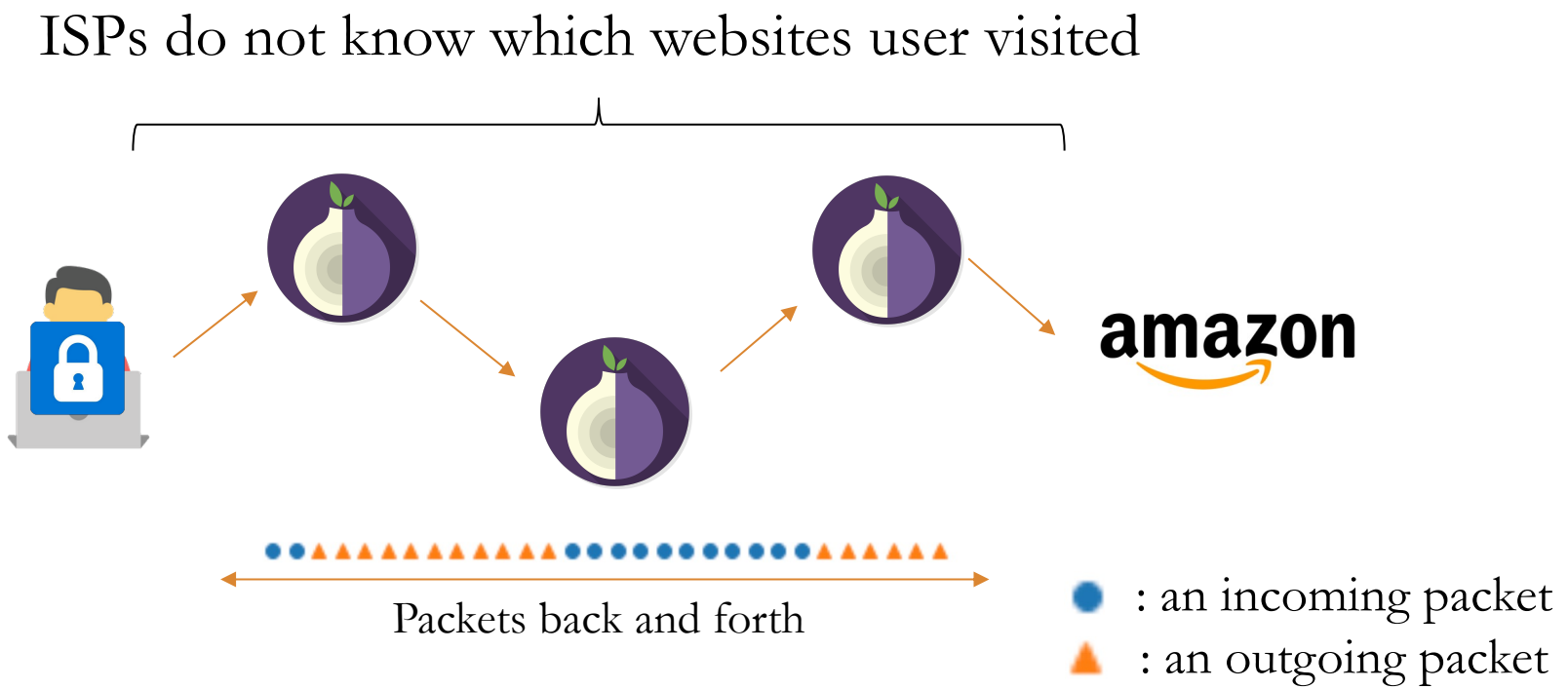
Patch-based Defenses against Web Fingerprinting Attacks

Shawn Shan
Arjun Nitin Bhagoji
Heather Zheng
Ben Y. Zhao

Tor and Website Fingerprinting



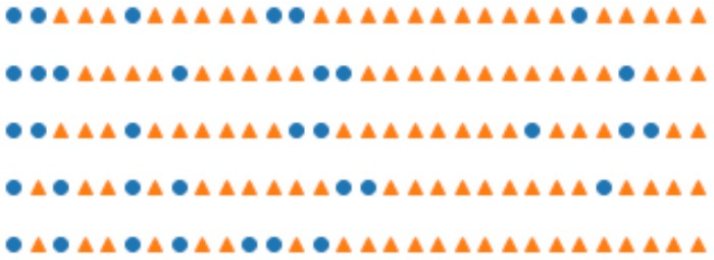
Anonymity tools provide privacy



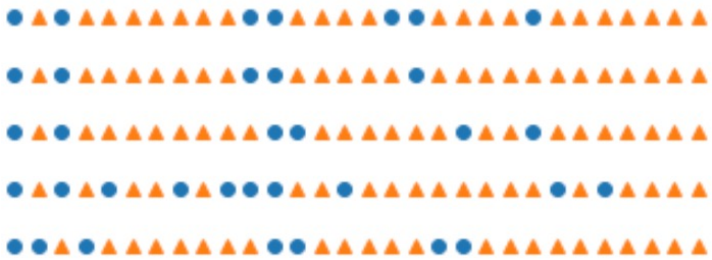
Can adversary infer website base on network traces?

Tor and Website Fingerprinting

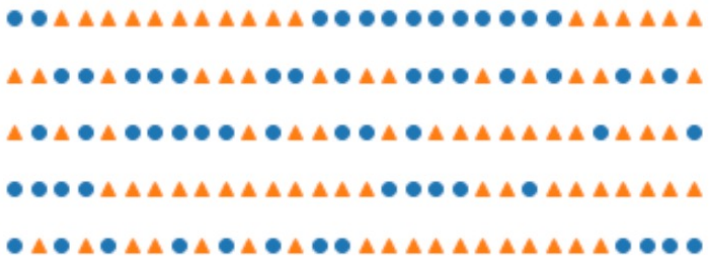
First 30 packets in the connection



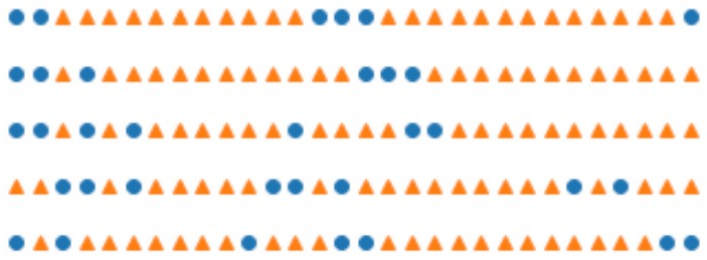
Amazon



YouTube



Wikipedia



LinkedIn

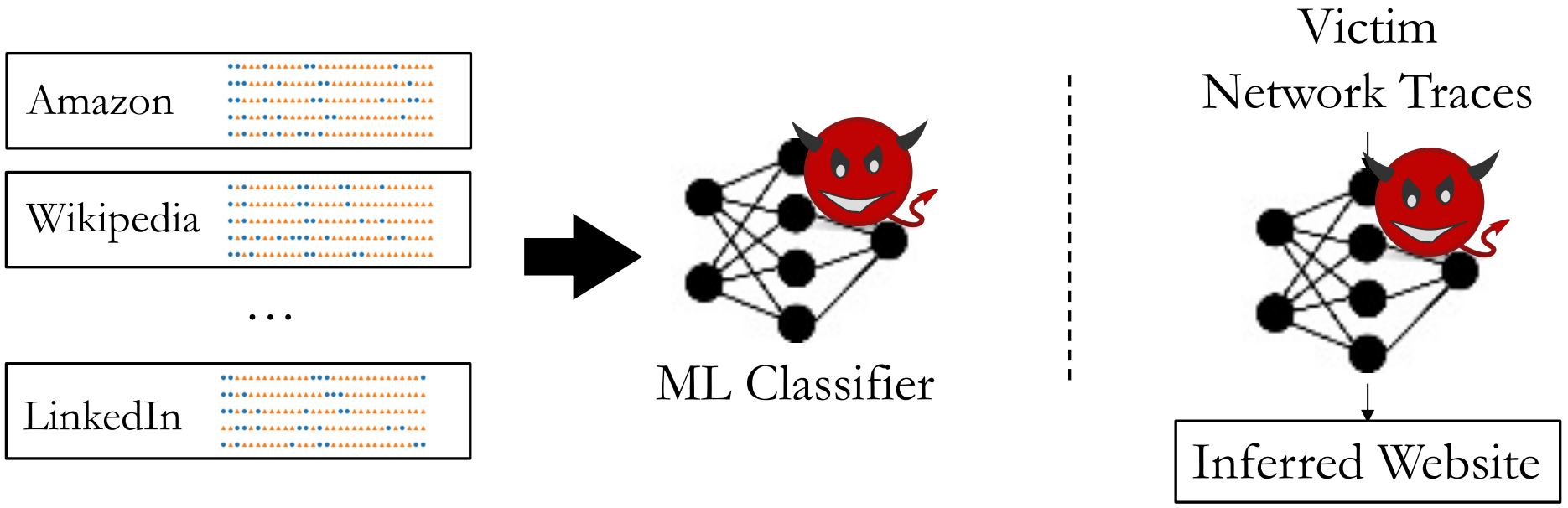
● : incoming packets

▲ : outgoing packets

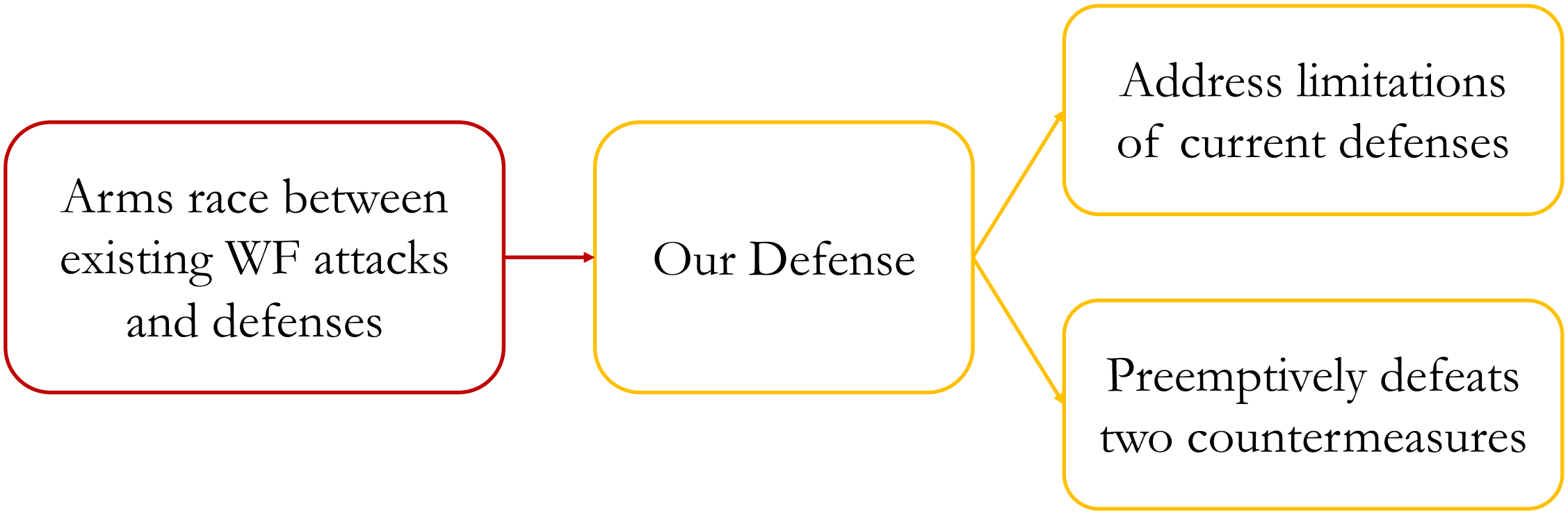
Packets are **encrypted** and padded to **same size**

Website Fingerprinting (WF) Attack

Leverage machine learning to classify websites



Outline


Arms race between existing WF attacks and defenses
Our Defense
Address limitations of current defenses
Preemptively defeats two countermeasures

Arms Race between Attacks and Defenses

Handcrafted features and ML Classifiers

Raw packet sequence

↓

Handcrafted Features

~**95**% Attack Success Rate

Arms Race between Attacks and Defenses

Handcrafted features and ML Classifiers

←

Make traces from different website similar to each other

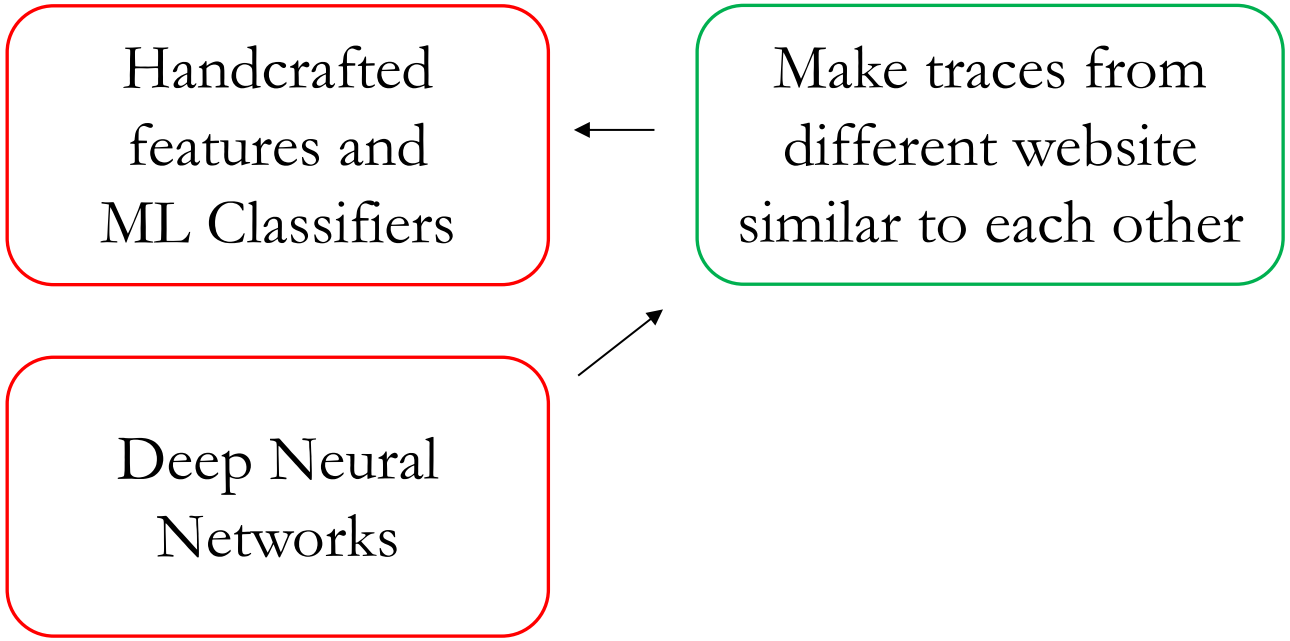
Defender: Insert **dummy packets**

2 ways to make trace look **similar**:

- Match Traces
- Inject Randomness

Reduce attack success to < 20%

Arms Race between Attacks and Defenses



Attacker: trains DNN on raw traces

- Raw input: sequence of 1 and -1
- RNN, CNN

~99% Attack Success Rate

↓

Adaptive attacker: Trains **DNN** on **defended traces**
(simulate with defense code)

~**97**% Attack Success Rate against **existing defenses**

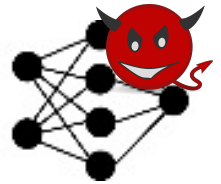
Arms Race between Attacks and Defenses

Handcrafted features and ML Classifiers

Deep Neural Networks

Make traces from different websi similar to each other

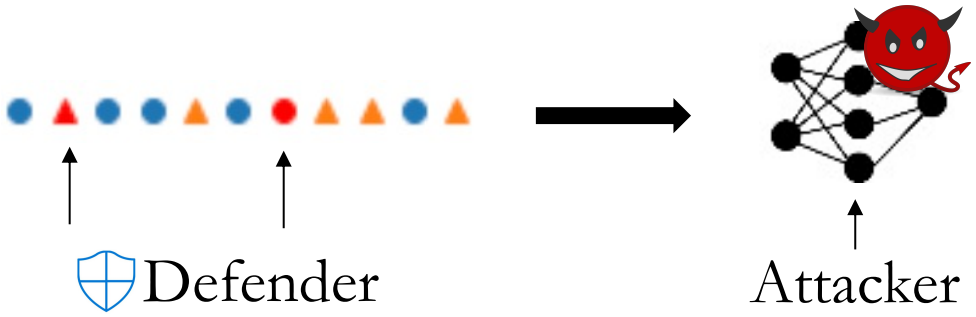
Adversarial perturbations as defense



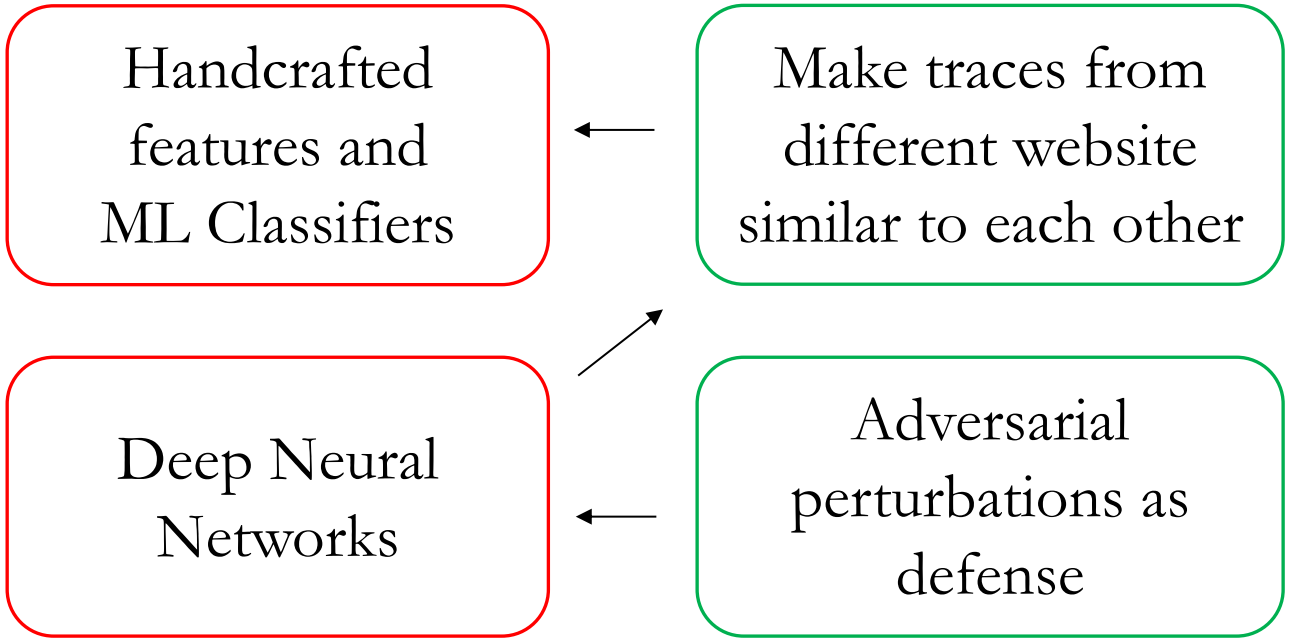
Trace of Website A: Website A

Adversarially perturbed: Website B

Flipped notation compared to traditional ML Security



Arms Race between Attacks and Defenses



Benefit of using adversarial perturbation:

- Target DNN models
- Challenging to avoid

Benefit of network setting:

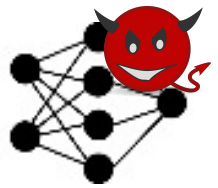
- Not noticeable
- Generous perturbation budget

Limitation of Adversarial Perturbation

Generating adversarial perturbation requires **the entire input**

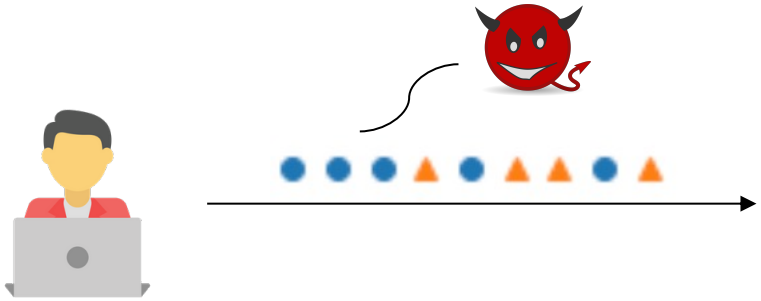
Original trace:

Perturbation optimization (e.g., PGD)



Defended trace:

No access to full input at defense time

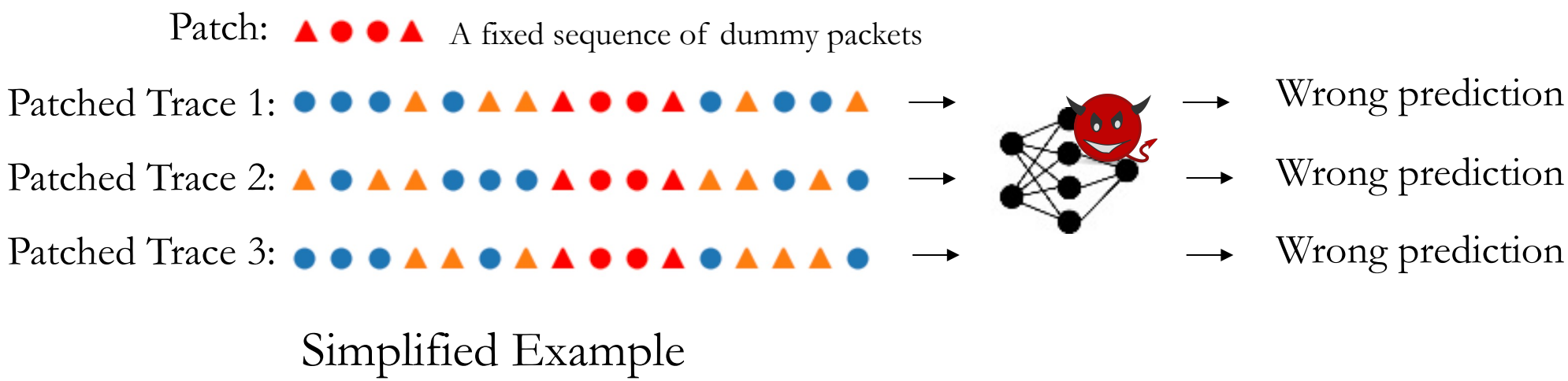


Too late for defense when full trace finished transmitting

Dolos: Patch-Based Defense

Use Universal Adversarial Patch

Universal on any input (input agonistic) → **Precompute** before defense time



Simplified Example

Two Potential Problems (Adaptive Attacks)

1

Defense Source Code
(Open sourced by Tor)

Recover defense patch ▲●●▲

Parameterized Randomness

2

Trace 1:

Trace 2:

Trace 3:

Runtime Randomness

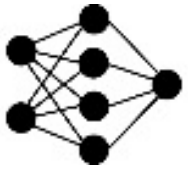
Intuition of Parameterized Randomness

1

Defense Source Code
(Open sourced by Tor)

Recover defense patch ▲●●▲

Retrain model on
defended traces



Solution

Goal: Even with source code, attacker cannot reproduce defense patch

Parameterized Randomness

Secret Key

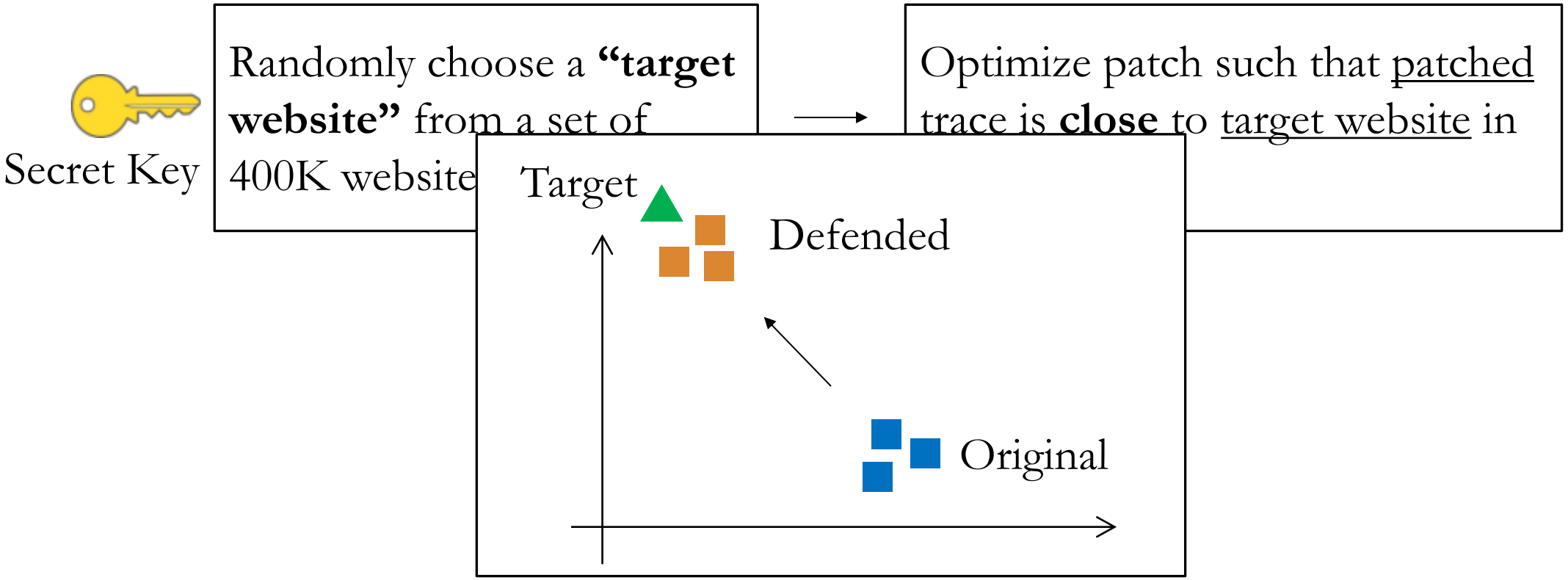
Key 1 ▲●●▲

Key 2 ●●▲▲

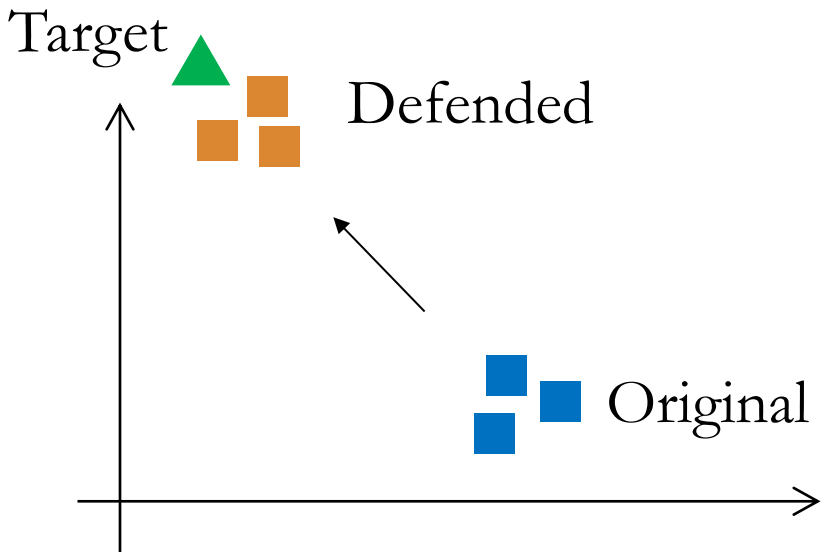
Key 3 ▲▲▲●

Design of Parameterized Randomness

Our implementation: encode **optimization direction** as the randomness



Patch Optimization



Patched trace current feature vector

Target feature vector

$$\mathbb{D}\left(\Phi(\Pi(p, x, s)), \Phi(x')\right)$$

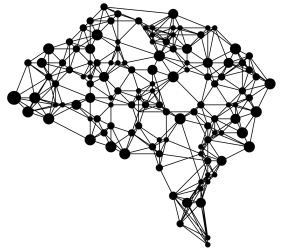
L_2 Distance

Evaluation Setup

User + **Dolos**

Secret Key

Original Traces +



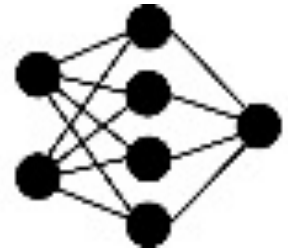
Well-trained

→

WF Attacker

Defended Traces

Trained using defended traces with a **different key**



Attack Classifier

Protection Success Rate: Percentage of defended traces misclassified by attacker's model

Overhead: number of dummy packets / total number packets

Evaluation Results

Defending Dataset	Defender's Feature Extractor	K-NN	K-FP	CUMUL	DF	Var-CNN
Sirinam						
Rimmer						

Dataset with 100 websites

Dataset with 900 websites

Evaluation Results

Architecture-Training Dataset

3 non-DNN Attacks

2 DNN Attacks

Defending Dataset	Defender's Feature Extractor	K-NN	K-FP	CUMUL	DF	Var-CNN
Sirinam	DF-Sirinam					
	DF-Rimmer					
	VarCNN-Sirinam					
	VarCNN-Rimmer					
Rimmer	DF-Sirinam					
	DF-Rimmer					
	VarCNN-Sirinam					
	VarCNN-Rimmer					

Dataset with 100 websites

Dataset with 900 websites

Evaluation Results

Architecture-Training Dataset

3 non-DNN Attacks

2 DNN Attacks

Defending Dataset	Defender's Feature Extractor	K-NN	K-FP	CUMUL	DF	Var-CNN
Sirinam	DF-Sirinam	98%	98%	97%	97%	96%
	DF-Rimmer	97%	98%	96%	95%	97%
	VarCNN-Sirinam	97%	98%	95%	94%	96%
	VarCNN-Rimmer	97%	98%	97%	95%	95%
Rimmer	DF-Sirinam	98%	97%	96%	96%	97%
	DF-Rimmer	97%	97%	97%	95%	97%
	VarCNN-Sirinam	98%	98%	97%	95%	97%
	VarCNN-Rimmer	98%	98%	98%	96%	98%

Dataset with 100 websites

Dataset with 900 websites

> 95% Protection success rate at 30% overhead

Comparison to Existing Defenses

All attack models are trained on defended traces

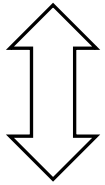
Defense Name	Overhead	DF	Var-CNN
WTF-PAD	54%	10%	11%
FRONT	80%	34%	31%
Mockingbird	52%	69%	73%
UAPs	30%	81%	73%
Dolos	**30%**	**96%**	**95%**

Universal perturbation → (w/o) parametrized randomness

Other Adaptive Attacks

Other adaptive attacks results in the paper

No approach exist that can effectively mitigate patches that are **non-consecutive** and have a **large perturbation budget**



Analytical result: With a sufficiently large budget, *no* classifier can correctly classify perturbed inputs

Conclusion

Dolos: Defending against WF attacks using adversarial patches with parameterized randomness

More interesting results in the paper

- Theoretical analysis
- Countermeasures