CellShift: RTT-Aware Trace Transduction for Real-World Website Fingerprinting

Rob Jansen, U.S. Naval Research Laboratory

Rob Jansen, PhD
Computer Scientist
Center for High Assurance Computer Systems
U.S. Naval Research Laboratory

The Network and Distributed System Security Symposium 2026
San Diego, CA, US
February 25th, 2026

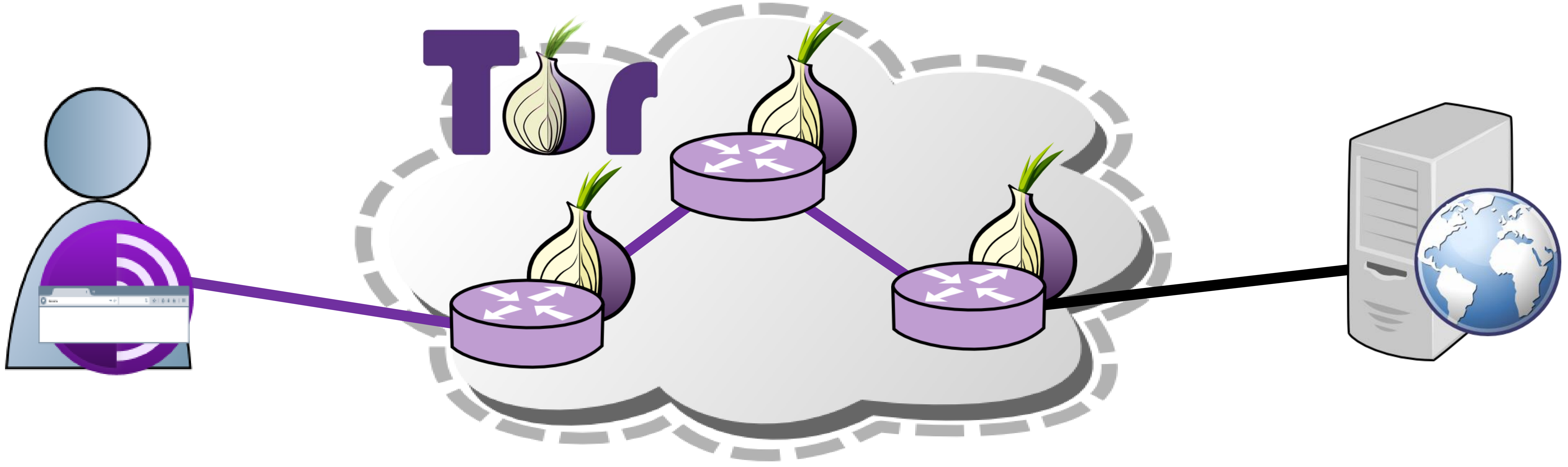
U.S. NAVAL RESEARCH LABORATORY

Anonymous Communication with Tor

- Separates *identification* from *routing*
- Provides unlinkable communication
- Promotes user safety and privacy online

Tor Browse Privately. Explore Freely.

Defend yourself against tracking and surveillance. Circumvent censorship.

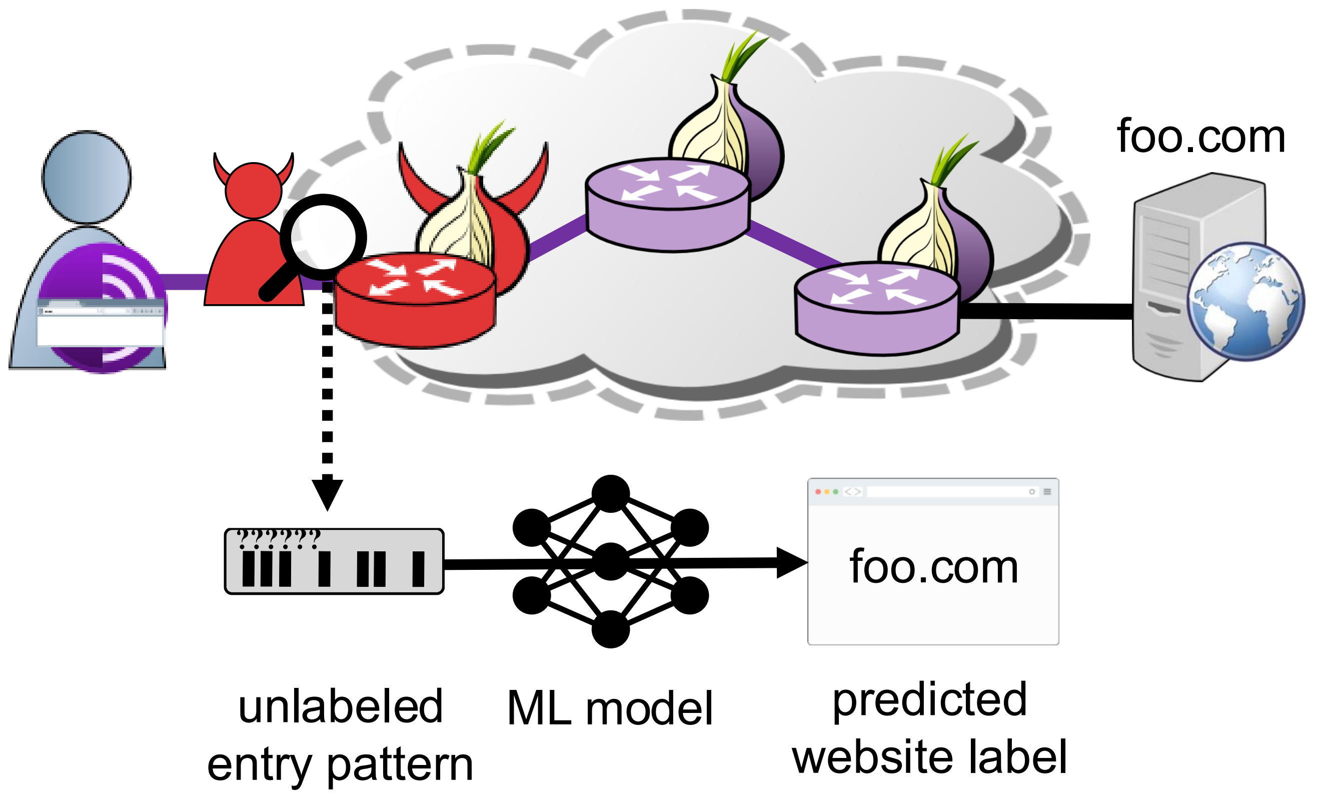


Website Fingerprinting Adversary Model

Website fingerprinting (WF) attacks can **link a source to its destination**, breaking Tor's anonymity

Adversary can:

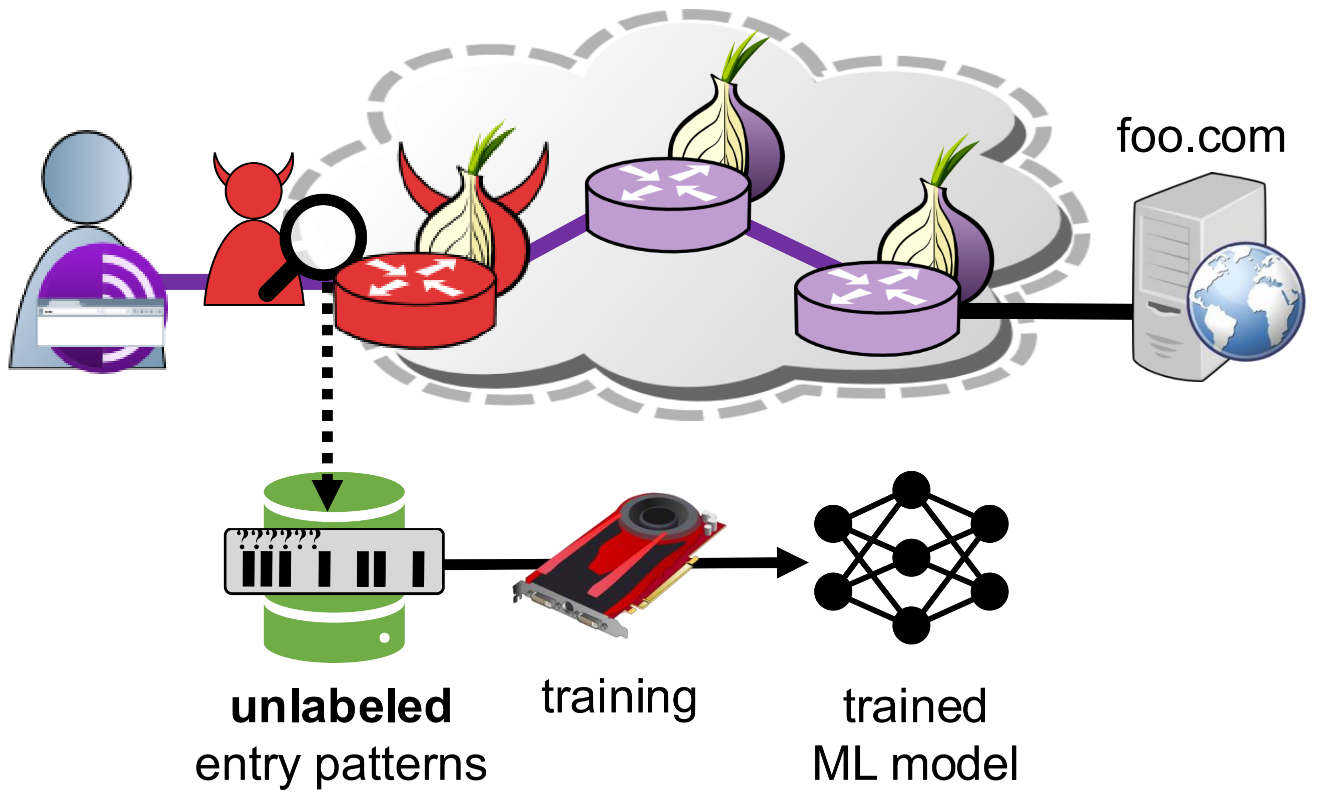
- Obtain entry-side vantage point
- Observe traffic patterns
- Predict website visited by user using **a trained ML model**



How Might Adversary Train ML Models?

Non-option: use **entry examples**

- Need **labeled** examples of patterns
- **Onion-encryption** hides labels
- Observed examples are **unlabeled**



U.S. NAVAL RESEARCH LABORATORY

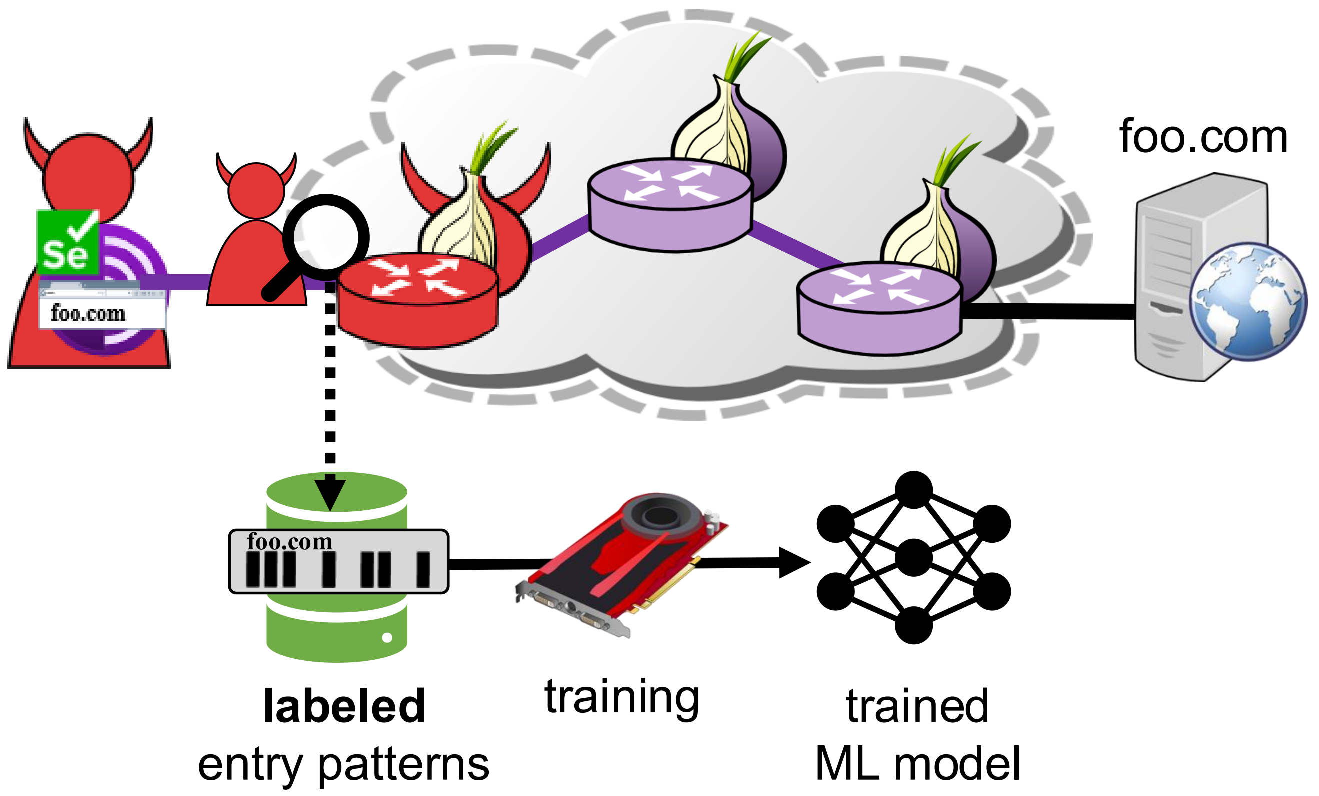
How Might Adversary Train ML Models?

Option 1: **synthetic data**

- Use **automated browser** (selenium) to **replicate users'** behavior/diversity
  - Usually by crawling frontpages of top sites…

Problem

- Modeling WF with synthetic user data **oversimplifies the ML task** [CCS'14, USENIX'22, PAM'26]
  - Browser version, configuration
  - URL choice, fetch order, usage of tabs
  - # of sites/pages, world size dynamics
  - Geo-location, concept drift
  - Tor network variation: relay churn, software versions, congestion, location…



U.S. NAVAL RESEARCH LABORATORY

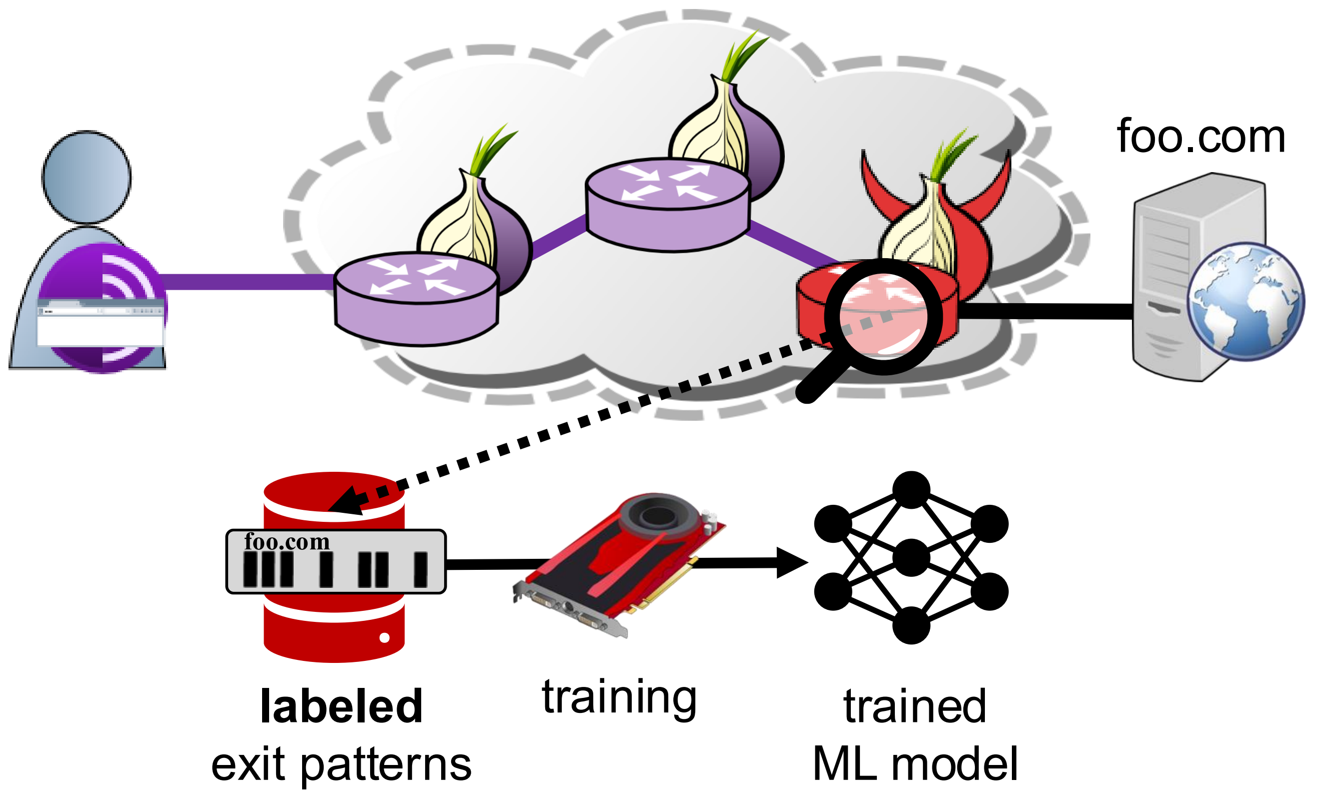
How Might Adversary Train ML Models?

Option 2: **real Tor user data**

- Run **exit relay**, observe traffic
  - Traffic patterns from **real Tor users**
  - Website **labels** observed in DNS requests

Problem

- Exit-entry position mismatch
  - **Train** on **exit** side, **predict** on **entry** side
  - Position “distortion” reduces performance
    - 5-18% [USENIX’22]
    - 17% median, 93% worse-case [WPES’24]



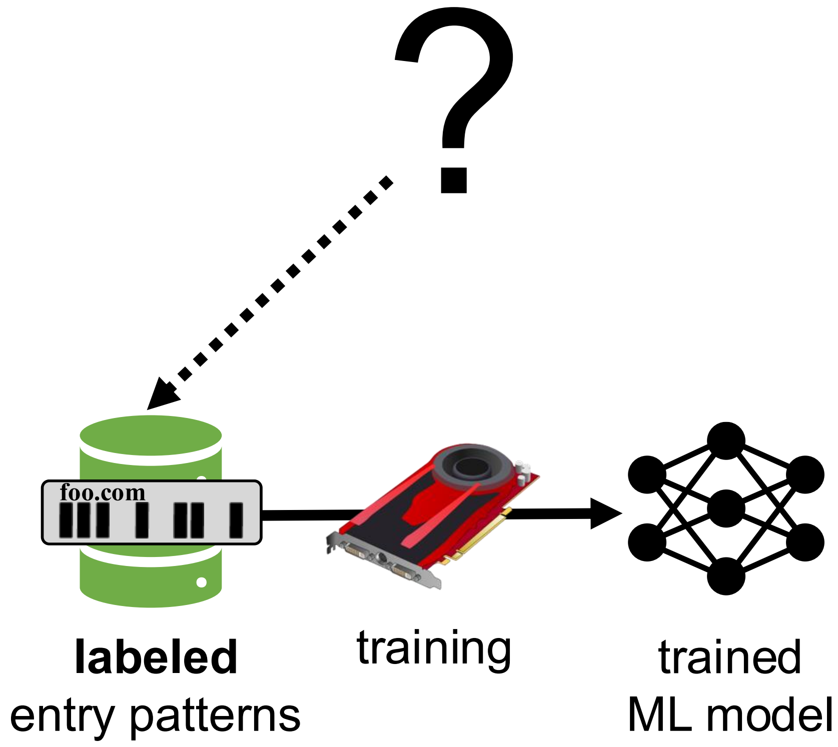
U.S. NAVAL RESEARCH LABORATORY

Our Research Direction: Key Insights

Key insights:

- If adversary would **test** on real user data, they should **train** on it too
- Easier to **mitigate** entry-exit distortion than **accurately replicate users**

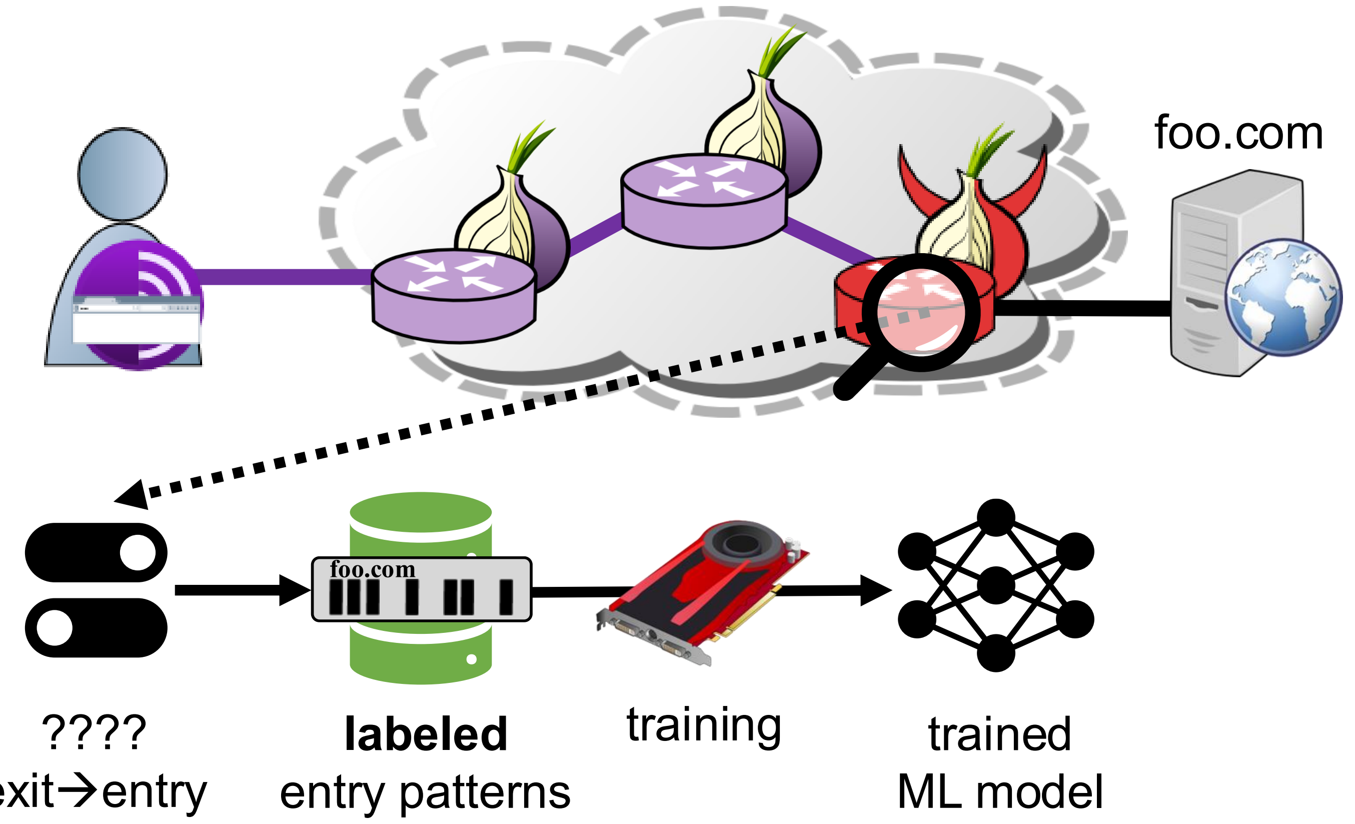
	Synthetic client	Real Tor exit
Ground-truth labels	✓	✓
Entry-side examples	✓	✗
Real Tor user data	✗	✓



Research Question and Goals

- Research question:
  - *Can exit→entry transduction efficiently improve classifier robustness to an out-of-distribution testing position?*
- Goals:
  - To understand the real-world threat of WF to **inform/prioritize defenses**
- Non-Goals:
  - Develop new WF attacks
  - Improve attacks to benefit adversary

	Synthetic client	Real Tor exit
Ground-truth labels	✓	✓
Entry-side examples	✓	✗ → ✓
Real Tor user data	✗	✓



U.S. NAVAL RESEARCH LABORATORY

CellShift: exit→entry Cell Trace Transduction

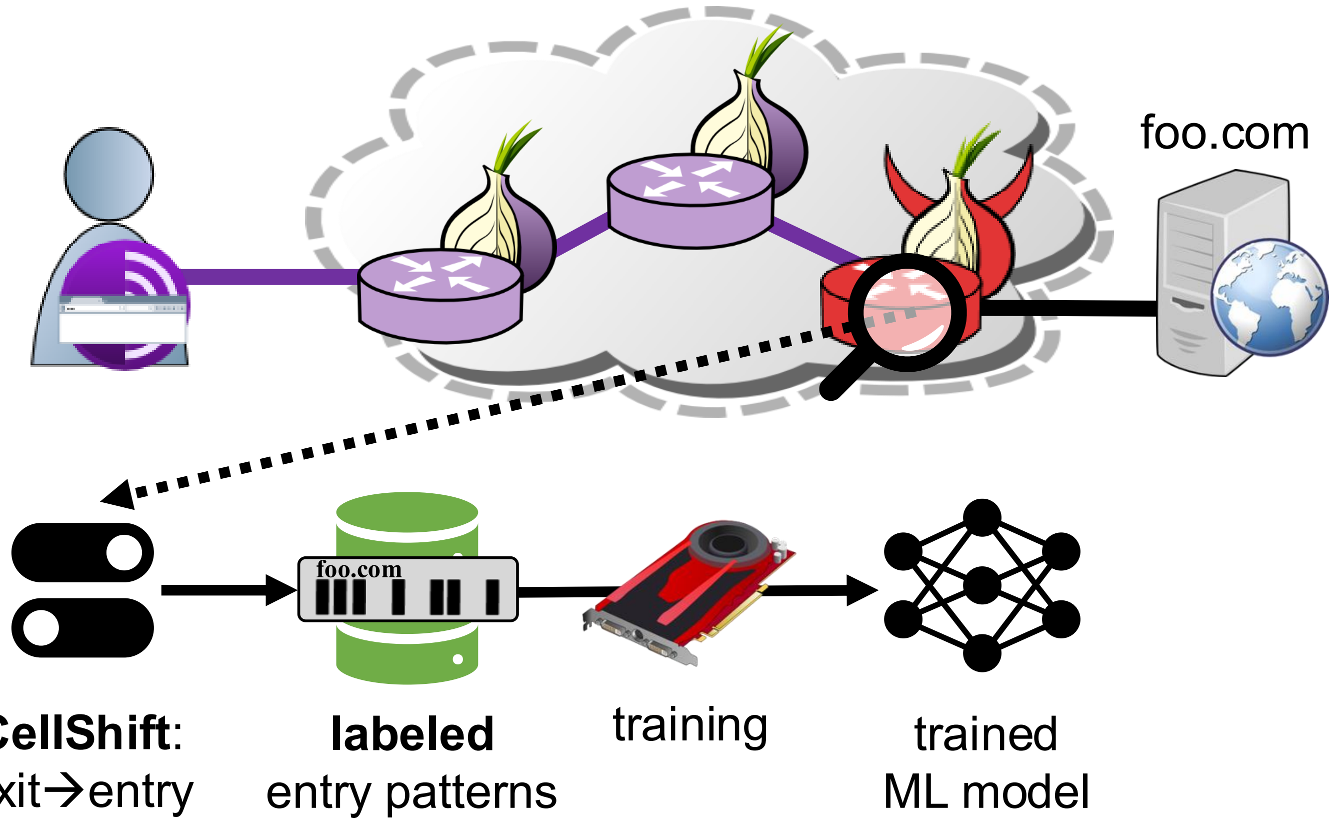
Introducing **CellShift**

- Transforms or “shifts” exit traces into entry traces

- Uses existing cell trace metadata
  - Extract **RTT estimates** from cell trace
  - Estimate e2e **latency** and **congestion**
  - Rewrite cell times to **simulate a shift** in position from exit to entry

- Uses only cell traces and … **MATH!**

	Synthetic client	Real Tor exit
Ground-truth labels	✓	✓
Entry-side examples	✓	✗ → ✓
Real Tor user data	✗	✓



How does CellShift work?

A **cell trace** is a sequence of:

- **Timestamp**:
  - when cell was observed
- **Direction**:
  - +1 client→server, -1 server→client
- **Command**:
  - type of protocol message

Example:

```
[
    (0.1, +1, CREATE),
    (0.5, -1, CREATED),
    (0.9, +1, BEGIN),
    (1.3, -1, CONNECTED),
    (1.3, -1, DATA),
    (1.3, -1, DATA),
    …
]
```

How does CellShift work?

A **cell trace** is a sequence of:

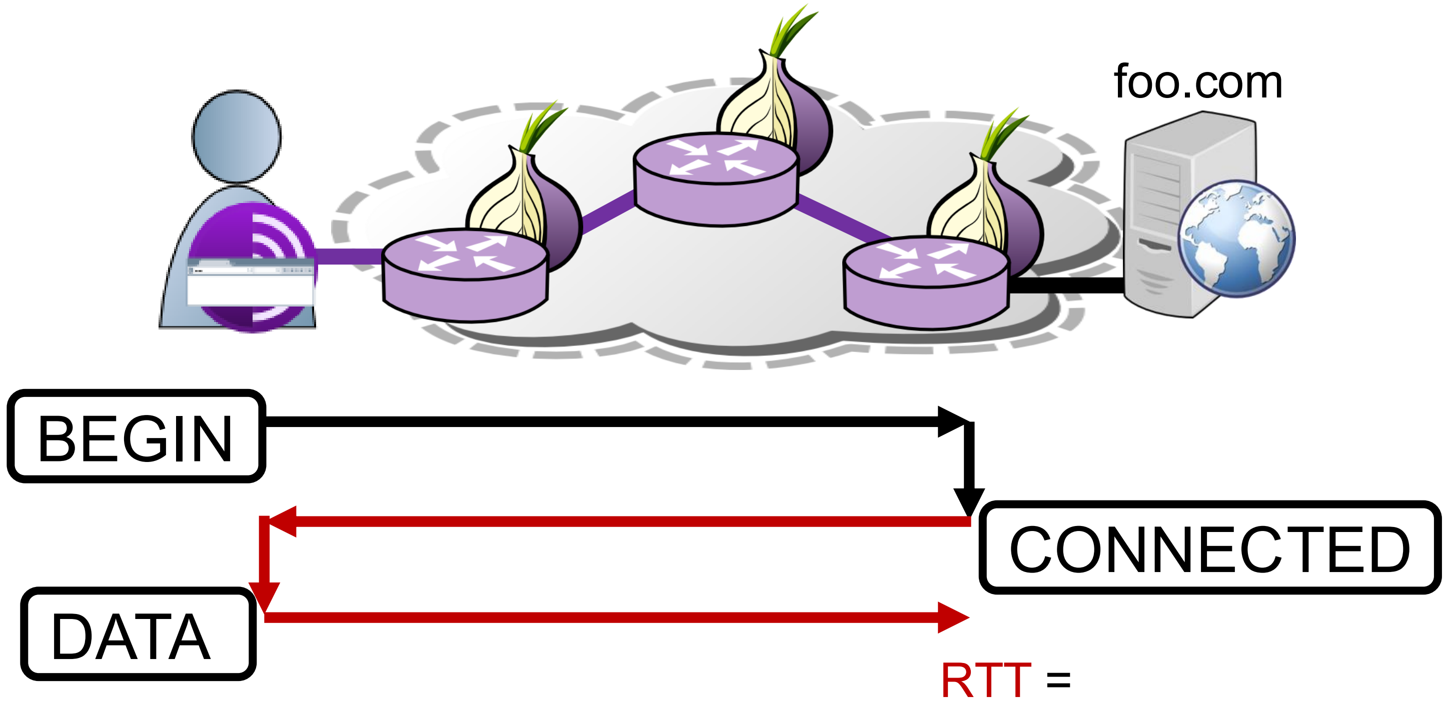
- **Timestamp**:
  - when cell was observed
- **Direction**:
  - +1 client→server, -1 server→client
- **Command**:
  - type of protocol message

Example:

```
[
    (0.1, +1, CREATE),
    (0.5, -1, CREATED),
    (0.9, +1, BEGIN),
    (1.3, -1, CONNECTED),
    (1.3, -1, DATA),
    (1.3, -1, DATA),
    …
]
```

Step 1: estimate circuit RTT from trace

- During server connection:
  - BEGIN → CONNECTED → DATA



RTT =
time received DATA - time sent CONNECTED

U.S. NAVAL RESEARCH LABORATORY

How does CellShift work?

A **cell trace** is a sequence of:

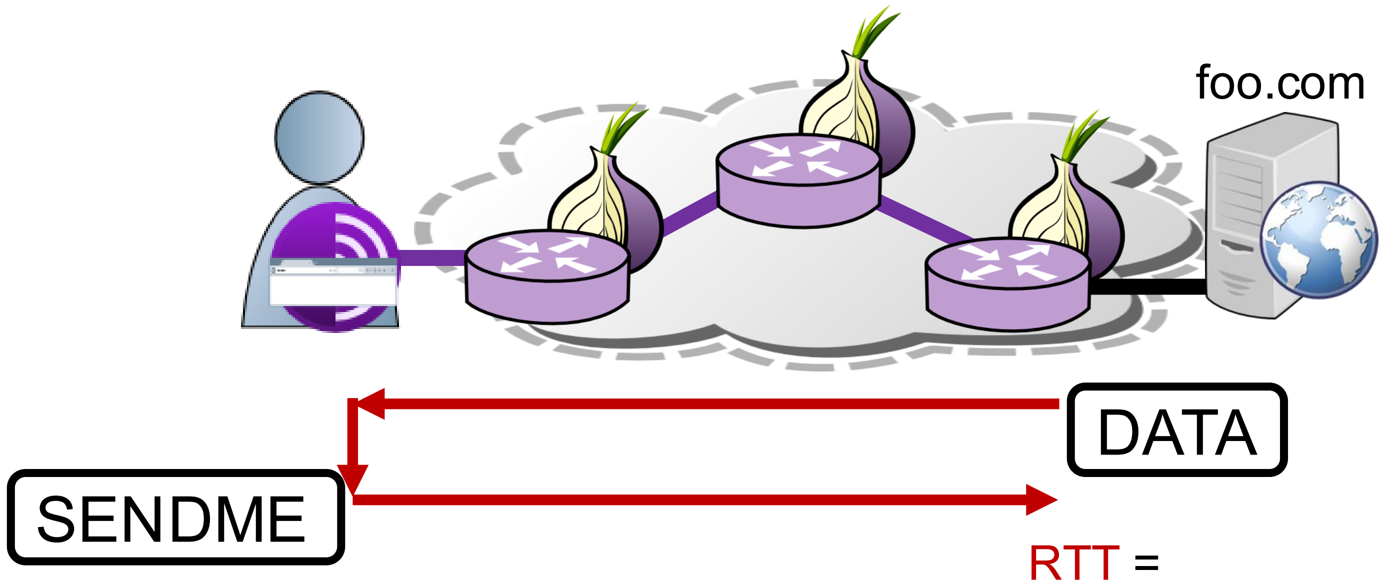
- **Timestamp**:
  - when cell was observed
- **Direction**:
  - +1 client→server, -1 server→client
- **Command**:
  - type of protocol message

Example:

```
[
    (0.1, +1, CREATE),
    (0.5, -1, CREATED),
    (0.9, +1, BEGIN),
    (1.3, -1, CONNECTED),
    (1.3, -1, DATA),
    (1.3, -1, DATA),
    …
]
```

Step 1: estimate circuit RTT from trace

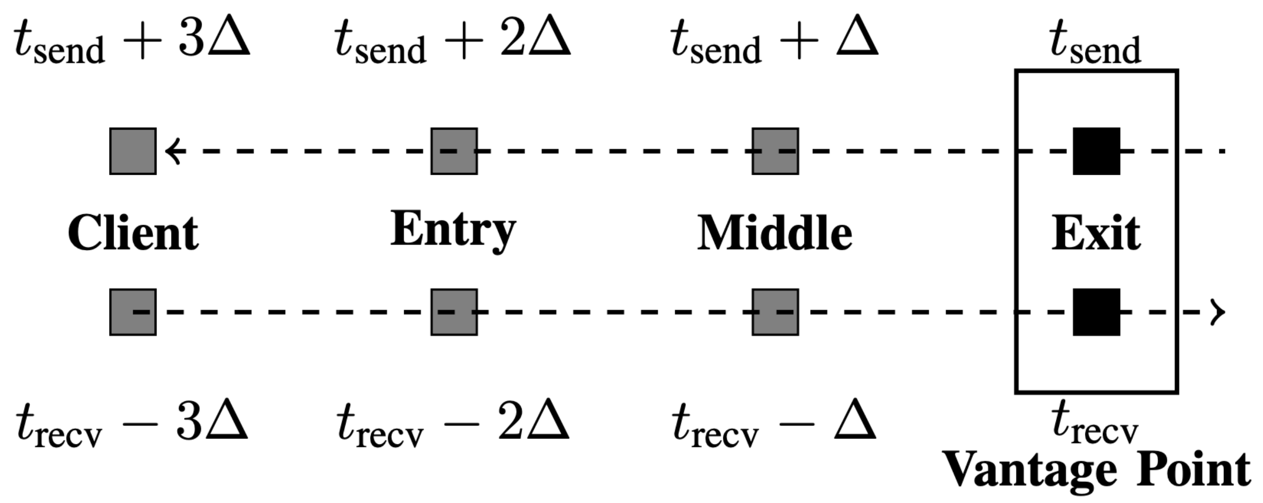
- During server connection:
  - BEGIN → CONNECTED → DATA
- During relay phase
  - Every 31st DATA → SENDME



RTT =
time received SENDME - time sent 31st DATA

How does CellShift work? (continued)

Step 2: Use RTTs to rewrite cell times



$t_{\text{send}} + 3\Delta$
$t_{\text{send}} + 2\Delta$
$t_{\text{send}} + \Delta$
t_{send}
Client
Entry
Middle
Exit
$t_{\text{recv}} - 3\Delta$
$t_{\text{recv}} - 2\Delta$
$t_{\text{recv}} - \Delta$
t_{recv}
Vantage Point

Figure 3: With an estimate of $\Delta = \frac{\text{RTT}}{6}$, a cell trace measured from an exit vantage point can be shifted to a new perspective by adjusting each cell's timestamp based on its direction.

Step 3: Re-sort the cells

$$\text{observed}_{\text{exit}} = \begin{array}{r}index\\ dir\\ time\end{array}\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ -1 & -1 & -1 & +1 & +1 & +1 \\ 0 & 10 & 20 & 50 & 60 & 70 \end{bmatrix}$$

$$\text{shifted}_{\text{entry}} = \begin{array}{r}index_{\text{old}}\\ index_{\text{new}}\\ dir\\ time\end{array}\begin{bmatrix} 3 & 4 & 5 & 0 & 1 & 2 \\ 0 & 1 & 2 & 3 & 4 & 5 \\ +1 & +1 & +1 & -1 & -1 & -1 \\ 10 & 20 & 30 & 40 & 50 & 60 \end{bmatrix}$$

Extensions for Augmented Training

Split **RTT estimates** into:

1. Propagation delay = RTT_{min}
2. $Congestion_i = RTT_i - RTT_{min}$

Can then be applied to *other* traces to produce **new trace variations** that **augment** the original trace

Producing augmented traces:

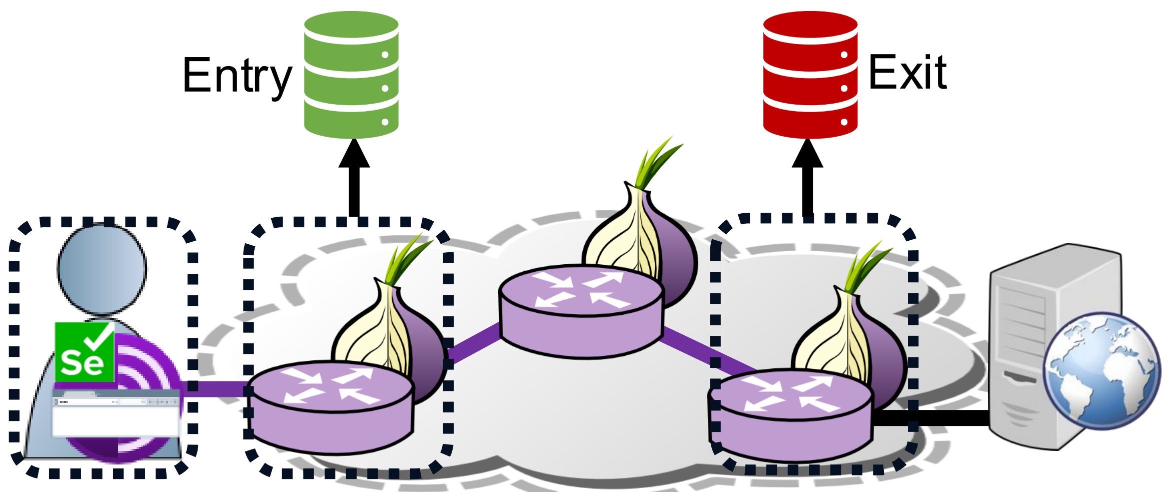
- **Simulates** loading trace through a *different path of relays*
- **Step1:** Create propagation delay **P** and congestion **C** distributions using the entire dataset
- **Step 2:** For each trace, adjust times by sampling from **P** and **C** to produce **n** augmented traces

U.S. NAVAL RESEARCH LABORATORY

Evaluation: Trace Distance

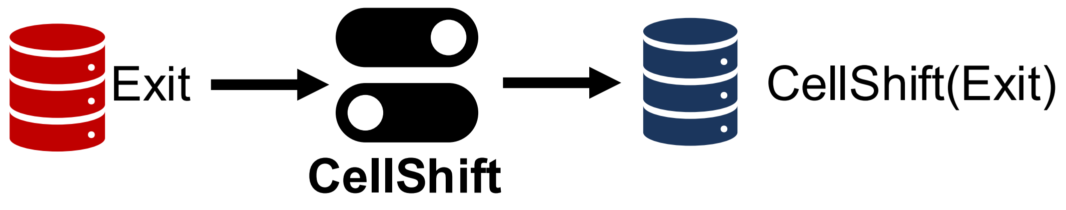
Collect **correlated** entry-exit traces

- Patched Tor to collect cell traces
- Pinned entry+exit on each circuit
- Fetch URLs multiple times each



patched tor to collect cell traces

Transduce exit→entry with CellShift

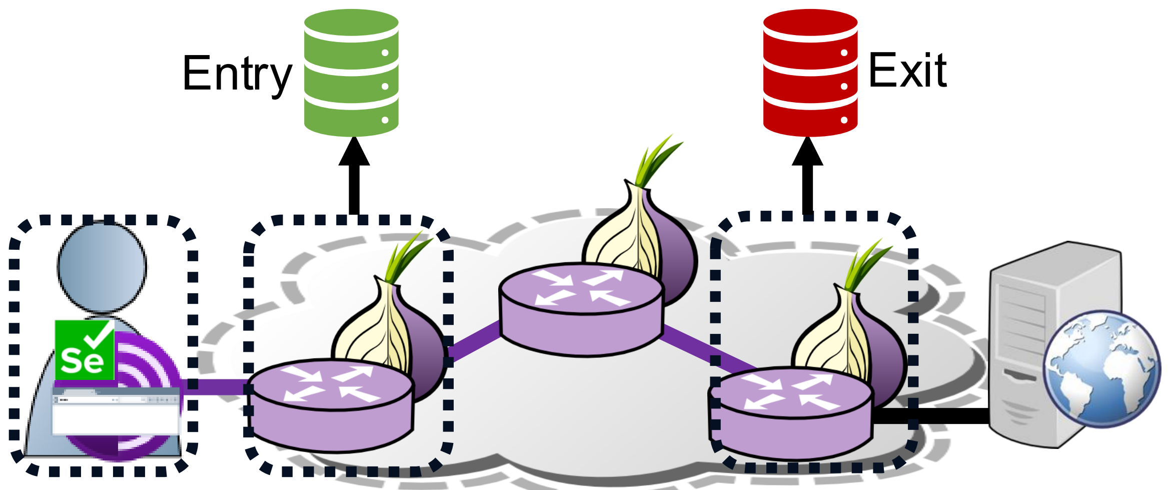


U.S. NAVAL RESEARCH LABORATORY

Evaluation: Trace Distance

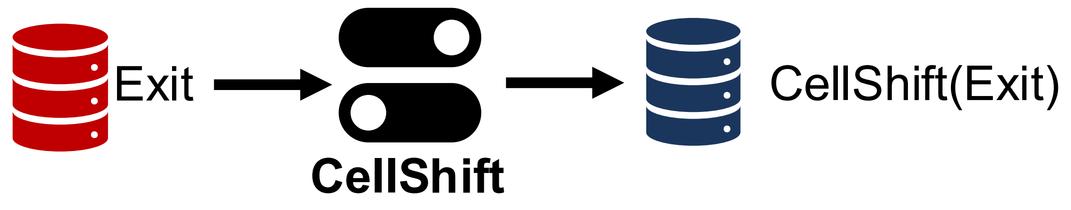
Collect **correlated** entry-exit traces

- Patched Tor to collect cell traces
- Pinned entry+exit on each circuit
- Fetch URLs multiple times each



patched tor to collect cell traces

Transduce exit→entry with CellShift



Compute **distance** to **Entry**

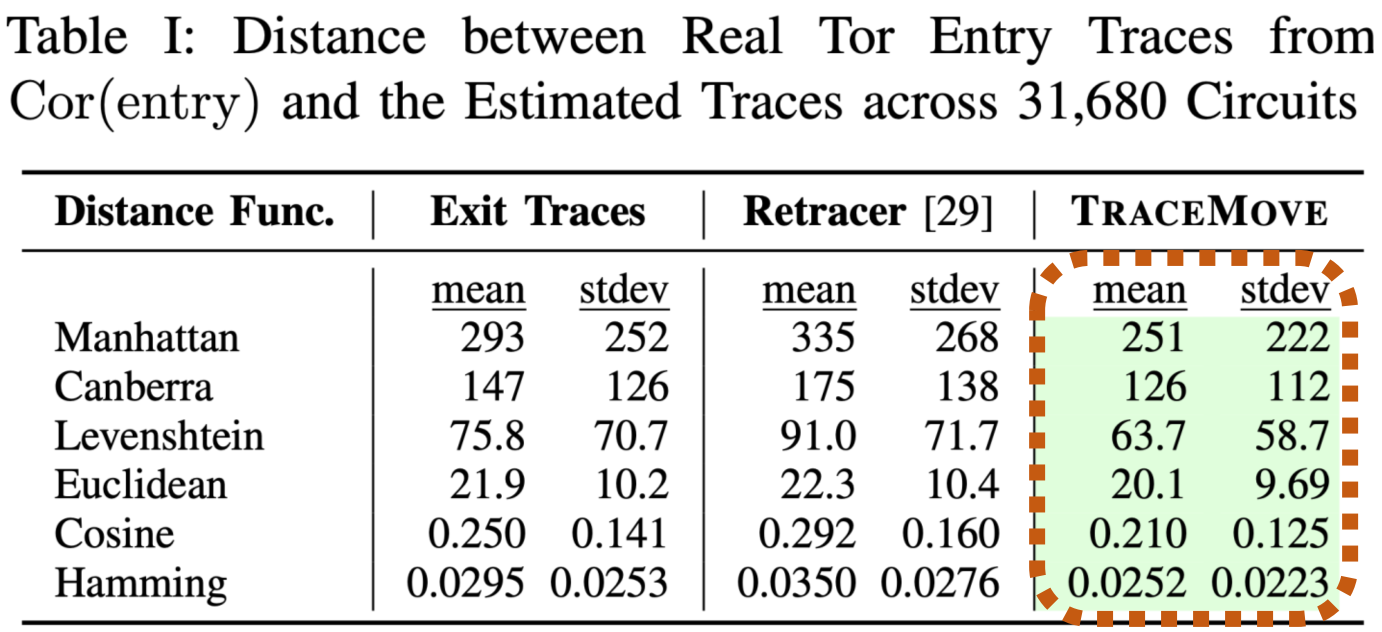


Table I: Distance between Real Tor Entry Traces from Cor(entry) and the Estimated Traces across 31,680 Circuits

Distance Func.	**Exit Traces**		**Retracer** [29]		**TraceMove**	
	mean	stdev	mean	stdev	mean	stdev
Manhattan	293	252	335	268	251	222
Canberra	147	126	175	138	126	112
Levenshtein	75.8	70.7	91.0	71.7	63.7	58.7
Euclidean	21.9	10.2	22.3	10.4	20.1	9.69
Cosine	0.250	0.141	0.292	0.160	0.210	0.125
Hamming	0.0295	0.0253	0.0350	0.0276	0.0252	0.0223

CellShift reduced distance across all tested distance functions

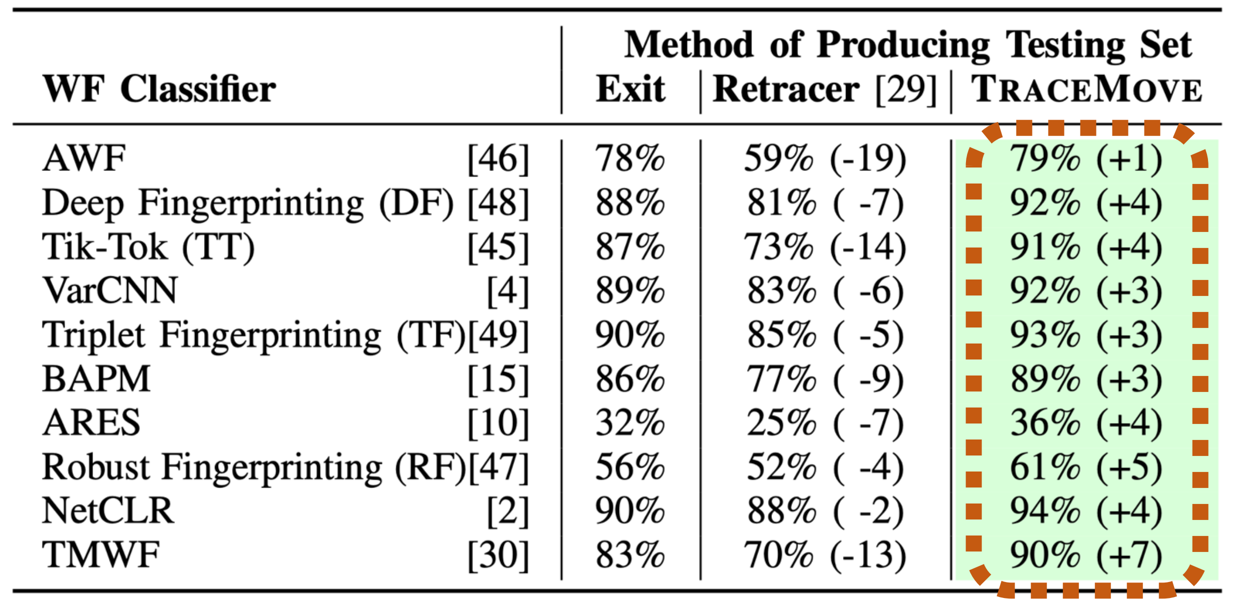
U.S. NAVAL RESEARCH LABORATORY

Evaluation: Closed-World WF with Synthetic Data

Hold **training** set constant, vary **testing**

- Synthetic Entry/Exit data [WPES'24]

Table II: Classifier Accuracy when Training on Tor(entry) in a Multi-Class Closed-World WF Experiment



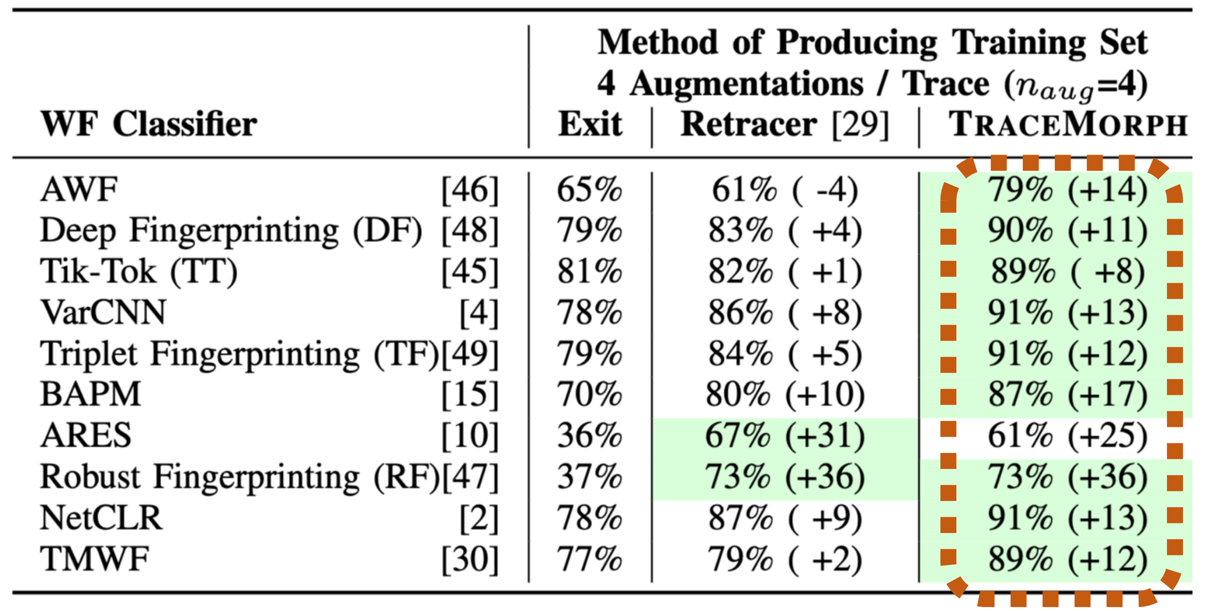
WF Classifier		Method of Producing Testing Set		
		Exit	Retracer [29]	TRACEMOVE
AWF	[46]	78%	59% (-19)	79% (+1)
Deep Fingerprinting (DF)	[48]	88%	81% (-7)	92% (+4)
Tik-Tok (TT)	[45]	87%	73% (-14)	91% (+4)
VarCNN	[4]	89%	83% (-6)	92% (+3)
Triplet Fingerprinting (TF)	[49]	90%	85% (-5)	93% (+3)
BAPM	[15]	86%	77% (-9)	89% (+3)
ARES	[10]	32%	25% (-7)	36% (+4)
Robust Fingerprinting (RF)	[47]	56%	52% (-4)	61% (+5)
NetCLR	[2]	90%	88% (-2)	94% (+4)
TMWF	[30]	83%	70% (-13)	90% (+7)

CellShift-based **TraceMove** improves WF accuracy across 10 tested classifiers by **1-7 pp**

Hold **testing** set constant, vary **training**

- Synthetic Entry/Exit data [WPES'24]

Table III: Classifier Accuracy when Testing on Tor(entry) in a Multi-Class Closed-World WF Experiment



WF Classifier		Method of Producing Training Set 4 Augmentations / Trace (n_{aug}=4)		
		Exit	Retracer [29]	TRACEMORPH
AWF	[46]	65%	61% (-4)	79% (+14)
Deep Fingerprinting (DF)	[48]	79%	83% (+4)	90% (+11)
Tik-Tok (TT)	[45]	81%	82% (+1)	89% (+8)
VarCNN	[4]	78%	86% (+8)	91% (+13)
Triplet Fingerprinting (TF)	[49]	79%	84% (+5)	91% (+12)
BAPM	[15]	70%	80% (+10)	87% (+17)
ARES	[10]	36%	67% (+31)	61% (+25)
Robust Fingerprinting (RF)	[47]	37%	73% (+36)	73% (+36)
NetCLR	[2]	78%	87% (+9)	91% (+13)
TMWF	[30]	77%	79% (+2)	89% (+12)

CellShift-based **TraceMorph** improves WF accuracy across 10 tested classifiers by **8-36 pp**

U.S. NAVAL RESEARCH LABORATORY

Evaluation: WF with Genuine Data

Genuine data from GTT23

- 13M traces from real Tor users, measured on Tor exits for 13 weeks
  - https://doi.org/10.5281/zenodo.10869889

Evaluated **closed** and **natural** world

Table IV: Classifier Accuracy when Testing on $\mathrm{GTT}^{\mathrm{cw}}_{100}$(test) in a Multi-Class Closed-World WF Experiment

WF Classifier		**Method of Producing Training Set**	
	OnlineWF [7]	**Retracer** [29]	**TraceMorph**
AWF [46]	33%	31% (-2)	52% (+19)
DF [48]	46%	46% (∼)	70% (+24)
VarCNN [4]	36%	34% (-2)	59% (+23)
TF [49]	41%	41% (∼)	60% (+19)
BAPM [15]	40%	34% (-6)	60% (+20)
ARES [10]	8%	7% (-1)	17% (+9)
RF [47]	11%	10% (-1)	16% (+5)
NetCLR [2]	42%	40% (-2)	67% (+25)
TMWF [30]	42%	41% (-1)	67% (+25)

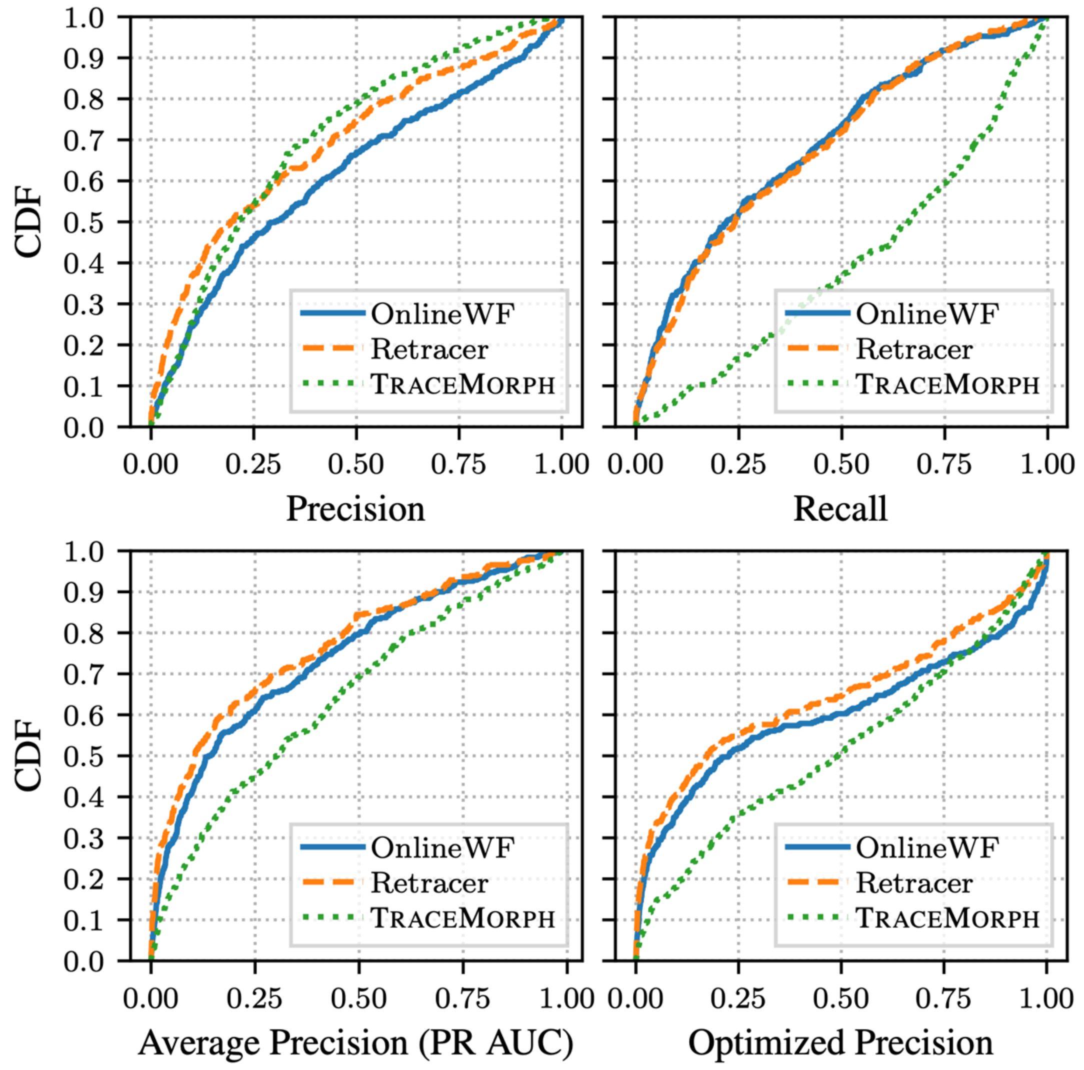


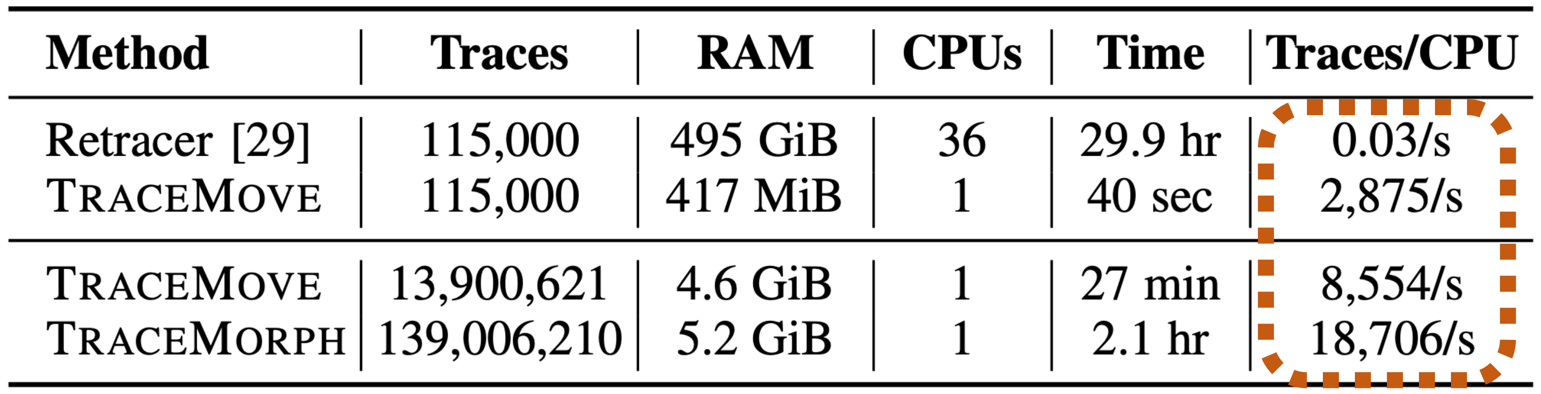
Figure 5: Classifier performance across per-website classifiers trained and tested in our natural-world setting using Dataset 4.

CellShift Transduction Performance

CellShift is **very** efficient

- Can process tens of millions of traces per hour per CPU
- Five orders-of-magnitude improvement relative to the state of the art
- Can handle typical Tor exit relay circuit load

Table V: Trace Transduction Performance



Method	**Traces**	**RAM**	**CPUs**	**Time**	**Traces/CPU**
Retracer [29]	115,000	495 GiB	36	29.9 hr	0.03/s
TraceMove	115,000	417 MiB	1	40 sec	2,875/s
TraceMove	13,900,621	4.6 GiB	1	27 min	8,554/s
TraceMorph	139,006,210	5.2 GiB	1	2.1 hr	18,706/s

CellShift: RTT-Aware Trace Transduction for Real-World Website Fingerprinting

Contributions

- **CellShift** for transducing exit→entry traces
- An exit→entry **distance** evaluation
- WF evaluation on **synthetic** data
- WF evaluation on **genuine** data
- **Open-source** Rust implementation

Future Work

- Use CellShift to:
  - Evaluate defenses against genuine data
  - Drive development of more informed defenses

Read Paper! **Download Code!**

Contact:
robert.g.jansen7.civ@us.navy.mil
robgjansen.com