IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 2 July 2024, accepted 18 October 2024, date of publication 28 October 2024, date of current version 5 November 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3487505*

RESEARCH ARTICLE

# DeepCoAST: Unveiling Split Trace Correlation to Counter Traffic Splitting Defenses

**GOUN KIM, HYEONJEONG KWAK, SUJIN KIM, YOUHEE PARK, JIHYEUN PARK, AND SE EUN OH, (Member, IEEE)**
Ewha Womans University, Seodaemun, Seoul 03760, Republic of Korea

Corresponding author: Se Eun Oh (seoh@ewha.ac.kr)

This work was supported in part by Ewha Womans University Research Grant of 2022; and in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korea Government (MSIT) (Artificial Intelligence Convergence Innovation Human Resources Development, Ewha Womans University) under Grant RS-2022-00155966.

**ABSTRACT** Despite its widespread adoption, Tor remains vulnerable to traffic analysis attacks, which enables both ends of the communication to be inferred by network-level adversaries. Notable examples of such attacks include website fingerprinting and end-to-end flow correlation attacks. Various defense techniques have been proposed to enhance the security of Tor against these threats, with traffic splitting defenses standing out as particularly effective. These defenses allow packets to be sent through multiple circuits without incurring additional bandwidth overhead, thereby limiting the amount of traffic observable by adversaries. In this paper, the potential of correlating split traces is thoroughly investigated using the proposed deep learning-based correlator called DeepCoAST. It is shown that properly merged split traces, upon correlated detection, could enable website fingerprinting attacks to effectively identify websites with high accuracy. Superior performance is demonstrated by DeepCoAST, achieving an Area Under the Receiver Operating Characteristic Curve (AUC) of 0.98 against 95 pairs of split traces generated by three traffic splitting defenses: TrafficSliver, HyWF, and CoMPS. This result highlights the need for further enhancement of traffic splitting Website Fingerprinting (WF) defense mechanisms against DeepCoAST-style attacks.

**INDEX TERMS** Anonymity, flow correlation attack, Tor, website fingerprinting.

## I. INTRODUCTION

Tor is among the most widely used anonymous networks, serving millions of daily users [1]. The aim of Tor is to safeguard connection anonymity by routing communications through three randomly selected proxies and employing three-layered encryption. One layer is decrypted by each proxy, ensuring that both ends of the communication are concealed, with each destination able to identify only the preceding location.

However, despite its protections, security challenges are encountered by Tor, particularly regarding the inference of online activities through traffic analysis [2], [3], [4], [5], [6], [7], [8]. This attack, known as Website Fingerprinting (WF), involves the identification of the destination (typically a website) based on communication between the client and the first Tor proxy (entry guard). New feature engineering techniques and machine learning algorithms are continuously developed by WF researchers to enhance fingerprinting capabilities, thereby accurately linking traffic to websites.

Another notable attack on Tor involves the correlation of ingress and egress traffic, which compromises user anonymity. This end-to-end flow correlation attack entails the matching of communication entering (ingress) and exiting (egress) the Tor network by the adversary. To combat this, various correlation functions have been developed, utilizing statistical methodologies such as Spearman's rank correlation [9] and cosine similarity [10].

Recent literature suggests that correlation detection can be enhanced through carefully designed feature extractors based on deep learning algorithms. Nasr et al. [11] developed 2-D Convolutional Neural Networks (CNNs) that stack traffic features extracted from ingress and egress traffic, with multiple kernels learning their local relationships. A higher

The associate editor coordinating the review of this manuscript and approving it for publication was Jenny Mahoney.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

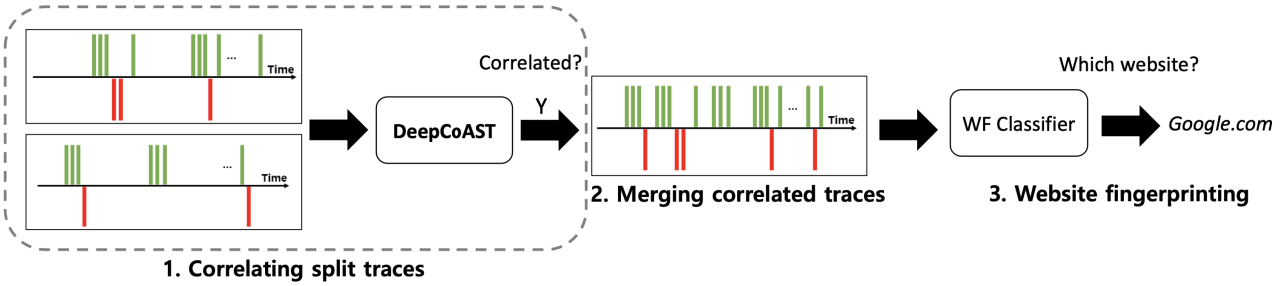



**FIGURE 1.** Traffic splitting defense mitigation scenario using DeepCoAST.

correlation detection rate was achieved by their approach using only a few hundred packets. Subsequently, Oh et al. [10] proposed generative deep learning models that generate more effective feature embedding vectors, demonstrating high correlation for correlated pairs of traces while exhibiting low correlation for uncorrelated pairs. Correlation detection performance was significantly improved by their new feature extractor model while requiring less computational cost than previous approaches [9], [11].

The fundamental vulnerability enabling both WF and end-to-end flow correlation attacks on Tor is the leakage of traffic metadata, such as packet timing and size information. This leakage allows the inference of the communication source and destination based on the correlation between the ingress and egress traffic of the same Tor connection.

To counter traffic analysis attacks on Tor, various defense mechanisms have been proposed, particularly targeting WF attacks, by padding Tor traces to obfuscate traffic patterns from network-level adversaries. However, significant bandwidth overhead is incurred by periodic padding [12], [13], which involves sending dummy packets at a constant rate, albeit effectively thwarting WF attacks. Alternatively, padding is selectively applied by lighter defense methods to critical packet locations, such as the front part of the trace [14], or to larger intervals between consecutive packets [15]. Despite reducing overhead, WF accuracy may be slightly increased by such defenses.

More recently, it has been demonstrated that sending packets through multiple paths is a more effective defense strategy, as the amount of traffic that can be monitored by adversaries observing only one path is reduced without incurring additional bandwidth overhead. Various methods have been developed to implement this approach. Multiple circuits, each routed through multiple entry guards, are utilized by TrafficSliver [16]. Multiple access points or internet service providers are leveraged by HyWF [17]. Connection migration, switching packets to multiple destination IP addresses, is employed by CoMPS [18]. In this paper, each traffic sent over each of multiple paths is referred to as a "split trace."

The multipath strategy has been proposed as a design improvement for Tor, aimed at enhancing the network's performance [19]. Although this approach does not specifically target defenses against WF attacks, it demonstrates the feasibility of splitting traffic across multiple circuits in Tor. Consequently, investigating the potential vulnerabilities introduced by traffic splitting defenses is crucial for further strengthening this approach, given its practicality, effectiveness, and zero bandwidth overhead. Addressing these vulnerabilities is essential for ensuring the security of Tor users. This research problem is both timely and important, as it addresses an underexplored yet significant threat to anonymity networks.

To this end, this paper presents a correlation model named DeepCoAST (Deep Correlation Attack for Split Traces). This model effectively detects correlations between split traces originating from the same Tor connection when traffic splitting defenses are deployed. By analyzing packet arrival times, the model merges split traces back into a single trace, reconstructing the original traffic pattern that would have been observed when using a single circuit. These merged traces allow WF attacks to regain sufficient traffic pattern recognition by utilizing the complete trace.

The capabilities of the previous flow correlator, DeepCoFFEA [10], are extended by DeepCoAST to effectively correlate split traces generated by three well-known splitting WF defenses: TrafficSliver [16], HyWF [17], and CoMPS [18]. The attack scenario with DeepCoAST is illustrated in Fig. 1.

To the best of our knowledge, the correlation between split traces induced by WF defenses using deep learning-based correlation models is investigated for the first time in this paper. An in-depth analysis of correlation features is included to propose a variant of recent WF features, termed correlated traffic features, along with various adjustments to DeepCoFFEA to develop DeepCoAST. Therefore, the scope of this paper is to develop an effective correlator for identifying correlated split traces, which are then merged (or reconstructed) into single-path traffic and subsequently fed into WF classification models, as outlined in the second and third steps of Fig. 1. The experimental details of our approach are provided in Section III-A.

It is noteworthy that while HyWF and CoMPS are not exclusively designed for the Tor network, these mechanisms were evaluated within the context of Tor. In this scenario, communication is split by the Tor client through two Autonomous Systems (ASes) before entering a Tor bridge. Therefore, throughout the paper, the terms "circuit" and "path" are used interchangeably.

The contributions of the paper are summarized as follows:

- A novel approach to counter traffic splitting defenses is proposed using the DeepCoAST model to learn the correlations between split traces. This represents the first correlation methodology specifically designed to attack traffic splitting defenses on Tor. The resulting correlator yields an AUC of 0.98, correctly correlating 95 pairs of split traces.
- To achieve such robustness, DeepCoAST is developed by adapting DeepCoFFEA as a correlator to detect the correlation between split traces. This transition requires an in-depth investigation of model architectures and feature explorations. Our implementation clearly shows the gap between DeepCoFFEA and DeepCoAST. First, DeepCoAST is based on a unified embedding network model, while DeepCoFFEA relies on two separate embedding models. Second, DeepCoAST utilizes intermediate-level granularity trace features, such as the number of packets per timeslot, whereas DeepCoFFEA uses fine-grained traffic features, such as packet timing and size information.
- WF features widely used in previous literature [2], [3], [4], [10], [20] are thoroughly explored, resulting in the proposal of a variant of a recent feature called Traffic Aggregation Matrix (TAM) [20] to enhance the split trace correlator. This exploration is supported by references to prior works such as $k$-NN [2], $k$-FP [3], Deep Fingerprinting [4], DeepCoFFEA [10], and Robust Fingerprinting [20].
- Our findings underscore the importance for security researchers to enhance current traffic- splitting defenses, given the substantial leakage of correlated traffic patterns.

While defenses involving two-path settings with adversaries monitoring two split traces are examined in this paper, it is important to note that TrafficSliver and CoMPS support more than three paths. However, the primary goal of this paper is to investigate the potential of existing flow correlation models to be extended to detect the correlation between two split traces and, in turn, undermine traffic splitting defenses. Possible extensions to correlate more than three split traces are also discussed in Section VI-C.

The organization of the paper is as follows: The background is discussed in Section II. The DeepCoAST is detailed in Section III. The feature analysis for effective DeepCoAST attacks against traffic splitting defenses is presented in Section IV. The details of the experiments, including the evaluation metrics, are provided in Section V. The performance of the proposed model is evaluated and the results are discussed in Section VI. Finally, the paper is concluded in Section VII.

## II. BACKGROUND

In this section, a review of prior literature on end-to-end flow correlation attacks, WF attacks, and defenses, which served as significant sources of inspiration for this paper, is presented.

### A. FLOW CORRELATION ATTACKS

The end-to-end flow correlation attack is recognized as a fundamental threat to Tor, as the egress and ingress traffic within the Tor network can be successfully correlated, thereby revealing both ends of the Tor connection. Traffic metadata, including packet timing and size information, has been explored by researchers to extract robust features that can be utilized in correlation using statistical methods, machine learning, or deep learning models. Despite the presence of considerable noise and perturbations in packet traces induced by the Tor network, this type of attack poses a significant threat to the anonymity of Tor users.

More recently, the potential of flow correlation attacks based on traffic analysis was demonstrated by Sun et al. in RAPTOR [9]. However, their methodology requires a substantial amount of traffic for monitoring. For instance, the attacks presented in their work demanded a 300-second-long traffic duration to exhibit satisfactory performance for a correlator based on Spearman's rank correlation algorithms.

Recognizing the impracticality of RAPTOR, a more efficient correlator, namely DeepCorr, was introduced by Nasr et al. [11], based on deep learning models. This model comprises multiple 2-D convolutional layers, stacking new features that include packet timing and size vectors extracted from both egress (traffic between exit nodes and destination servers) and ingress (traffic between clients and guard nodes) traffic within the Tor network. This novel approach significantly reduces the amount of traffic monitored to a few hundred packets per connection, from which traffic feature vectors are extracted.

However, both RAPTOR and DeepCorr demonstrate a drawback in their quadratic complexity. They require $n^2$ comparisons or stacked features, respectively, to identify the correlation between $n$ pairs of egress and ingress traffic on the Tor network. Moreover, this pairwise setting has led to a low base rate, $\frac{1}{n}$, rendering the flow correlation attack impractical in real-world scenarios. To demonstrate the feasibility of end-to-end flow correlation attacks on Tor in practice, there is a need for lower false positives.

Following previous literature, DeepCoFFEA was introduced by Oh et al. [10], employing a generative deep learning model fed by $n$ triplets to learn the correlation between $n$ pairs of egress and ingress traffic on the Tor network. Greater efficiency in both training and testing is achieved by this model. Additionally, an amplification technique is utilized by dividing each trace into multiple short-interval windows, evaluating each window separately, and aggregating results from multiple windows in an ensemble. This approach dramatically reduces the number of false positives, enhancing the practicality of the Tor flow correlation attack.

In this work, inspiration is drawn from DeepCoFFEA but a different objective is pursued. While DeepCoFFEA excels in detecting the correlation between egress and ingress

traffic in the Tor network, the extensibility of the flow correlation model in detecting the correlation between split traces generated by traffic splitting defenses is explored.

### B. WEBSITE FINGERPRINTING ATTACKS

WF attacks have been recognized as another fundamental threat to Tor and have been extensively studied by security researchers. Informative traffic patterns are revealed by Tor traffic metadata captured from the communication between the Tor client and entry guards, enabling the determination of the destination of Tor connections with high accuracy.

In earlier studies, traditional machine learning models were demonstrated to be effective as WF models when fed with carefully crafted features. For instance, diverse types of features, including cell and burst sequences, were utilized by $k$-NN [2]. A new type of feature, namely cumulative packet size sequence, was incorporated by CUMUL [21]. Furthermore, 150 hand-crafted features, which were well-suited for Random Forest classifiers, were investigated by $k$-FP [3].

More recently, to alleviate concerns about cumbersome feature engineering, researchers have turned to applying Deep Learning (DL) models to WF. A substantial WF dataset necessary for effectively training DL-based WF models was amassed by Automated Website Fingerprinting (AWF) [5]. Their work tailored Stacked Denoising Autoencoder (SDAE), 2-D CNNs, and Long Short Term Memory (LSTM) models to WF. At the same time, a robust WF model based on deep 1-D CNNs, called the Deep Fingerprinting (DF) model, was introduced by Sirinam et al. [4], showcasing its effectiveness in identifying a user's visited websites in both vanilla Tor traffic and defended Tor traffic. Subsequently, the DF model was expanded by Rahman et al. [22] by incorporating a novel feature set, Tik-Tok features. This feature set encompasses a sequence comprising both packet direction and packet arrival time information. Superior performance compared to the original DF model was demonstrated by their approach.

Despite the superior classification ability exhibited by DL-based WF models, the substantial amount of Tor traffic required poses a significant challenge, necessitating considerable efforts in regularly collecting extensive traces. To address this challenge, more advanced deep learning models were utilized by researchers [8], [23] to minimize the amount of required training data while still achieving comparable performance in WF compared to earlier WF models.

Following this, a Robust Fingerprinting (RF) model based on 2-D CNNs with a new category of WF features termed "Traffic Aggregation Matrix (TAM)" was introduced by Shen et al. [20]. TAM is constructed by dividing Tor traces into multiple short-interval window segments and aggregating packet counts per window. These window-based aggregated features effectively capture informative features even in padded or split traces, enabling the RF model to detect defended traces with high accuracy.

While this work is not directly focused on WF studies, an exploration of various WF features, including Tik-Tok and TAM, was conducted as part of the efforts to identify more effective correlated traffic features for DeepCoFFEA in detecting correlation between split traces. Notably, due to TAM's outstanding performance in identifying websites against TrafficSliver defenses, TAM was extended to be integrated into the DeepCoFFEA architecture. This extension is discussed in detail in Section IV.

### C. WEBSITE FINGERPRINTING DEFENSES

As WF attacks have evolved with the introduction of new features and models, effective defenses to safeguard the anonymity of Tor users have concurrently been developed by security researchers. In earlier efforts, regular padding-based defenses [12], [13], [24] were formulated, involving the periodic transmission of dummy packets to obfuscate the traffic pattern unique to the destination. Despite the success demonstrated in hindering the accuracy of WF models, a substantial amount of bandwidth overhead was incurred by these defenses due to the addition of dummy packets.

To mitigate the overhead to some extent, selective padding was introduced by WTF-PAD [15], where dummy packets were sent only to fill statistically larger gaps. Reduced overhead was achieved by this approach while maintaining a slightly higher WF accuracy.

After this, additional research aimed to lower WF accuracy while minimizing bandwidth overhead by mimicking representative traffic sequences during the padding of traces. An example of such an approach is Walkie-Talkie [25], constructed based on the half-duplex mode, which simulates one or more websites for each website to have them carry the same traffic pattern. The intention is to make them indistinguishable from each other, thereby achieving a balance between reduced WF accuracy and lower bandwidth overhead.

To counter the ability of WF attackers to monitor communication between the client and the entry guard through a single path (e.g., a single entry guard), traffic splitting defenses have been developed by researchers, involving the sending of packets across multiple network paths. For instance, TrafficSliver [16] achieves this by employing multiple entry guards, HyWF [17] divides communication across multiple Internet Service Providers, and CoMPS [18] separates traffic through connection migration.

Powerful performance against recent WF attacks has been demonstrated by these defenses while incurring zero bandwidth overhead.

In this paper, the specific focus lies on three splitting defenses—TrafficSliver, HyWF, and CoMPS—to identify potential vulnerabilities, particularly correlated traffic patterns between split traces of the same connection. Detailed insights into each of these three defenses are provided in Section II-E and the performance of the attack model against these defenses is evaluated in Section V.

### D. IOT INTRUSION DETECTION SYSTEMS

The important role of network traffic features in protecting the security of the Internet of Things (IoT) has been recognized.

The IoT comprises interconnected devices equipped with sensors, enabling data to be exchanged with other devices and subsystems. As the IoT ecosystem expands, the incidence of intrusions—unauthorized or malicious attempts to gain access and compromise data—has also increased. Consequently, the safeguarding of IoT systems by thwarting unauthorized access and enhancing overall security has made Intrusion Detection Systems (IDS) essential.

Numerous research [26], [27] have been conducted to propose IoT IDS based on various machine learning and deep learning models to understand malicious IoT traces based on statistical traffic information such as IP addresses, port numbers, protocol types, and specific application identifiers. In particular, the efficiency of the IDS process was explored by Nallakaruppan et al. [28] using various machine learning algorithms. An effective algorithm and framework were proposed by them to mitigate intrusions on the host side.

### E. OVERVIEW OF TARGETED TRAFFIC-SPLITTING WF DEFENSES

In this section, three traffic splitting defense mechanisms that are targeted by the adversary are detailed. Furthermore, the vulnerabilities that enable the detection of correlations between split traces are discussed.

#### 1) TrafficSliver

'As illustrated in Fig. 2a, a lightweight WF defense employing a traffic splitting strategy was introduced by Cadena et al. [16]. TCP packets are transmitted over multiple circuits between the client and middle node through the use of multiple entry guards. Subsequently, individual TCP streams are merged at the middle node.

This defense limits the adversary's ability to observe only partial traffic along one circuit, thereby restricting exposure to potentially identifiable traffic patterns. Consequently, its effectiveness and efficiency in protecting against WF on Tor are recognized, as no bandwidth overhead is incurred.

Traffic splitting at the application layer was also demonstrated by Cadena et al., where HTTP requests are sent over multiple circuits by configuring browser proxies with different Tor circuits. However, the exclusive focus of this paper is on TrafficSliver in the network-level setting, where TCP streams are sent over multiple entry nodes.

The number of entry guards was varied, and multiple splitting schemes including round robin, random splitting, weighted random (WR), and batched weighted random (BWR) were explored. The round-robin scheme simply transitions to the subsequent circuit with each packet. The random splitting scheme randomly chooses the path for each packet. In the WR scheme, each entry guard is assigned a probability $w$ derived from a Dirichlet distribution, with these probabilities summing up to 1. These probabilities influence the selection of the path when sending packets. In the BWR scheme, packets are sent in batches comprising $n$ packets to the designated entry node. Additionally, for each batch, the value of $n$ is re-sampled.

The exclusive focus of this paper is on the BWR scheme, acknowledged as the most effective configuration. Notably, their experimental results indicated that selecting a value for $n$ within the range of 50 to 70 resulted in the highest security. This is one of the settings used to configure TrafficSliver in Section V.

Despite the BWR scheme introducing randomness to the selection of circuits and the number of packets to be sent, correlations among split traces were observed. Specifically, correlations in the inter-packet arrivals of split traces were identified. This refers to the intervals between consecutive packets, packet sizes, and aggregated packet statistics computed from each fixed-time interval. Since packets are transmitted asynchronously across multiple circuits, upon detecting correlations between split traces, correlated split traces can be effectively merged, reconstructing the full trace as if it were collected using a single entry guard.

#### 2) HyWF

A HyWF that entails sending packets across two networks provided by two Internet Service Providers (ISPs) or two different access points such as Wi-Fi and cellular networks was proposed by Henri et al. [17]. The objective of this strategy is to eliminate some of the packets by redirecting them over another network.

After various splitting schemes were explored, a multipath scheduler that dynamically adjusts both the number of packets sent over the network and the probability of selecting a particular network was introduced.

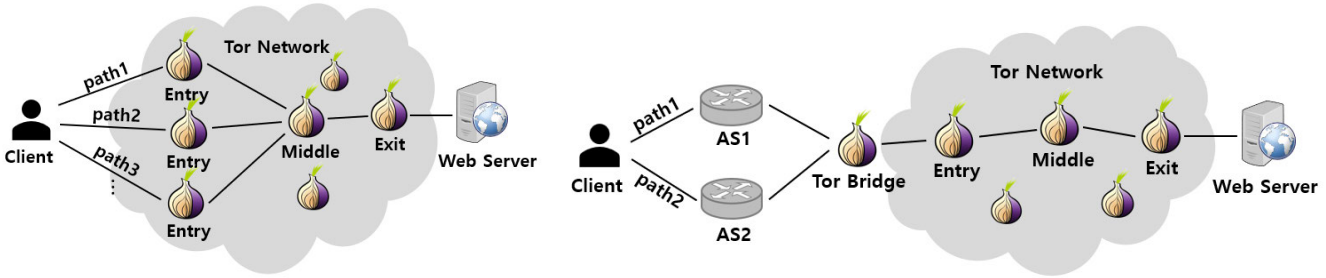
The number of consecutive packets, denoted as $n$, to be sent over the network is selected from a geometric distribution with a mean value of $n_{cons}$. Subsequently, the probability $p$ to choose one network is drawn from the Bernoulli distribution, while the probability of selecting the other is $1 - p$. This approach incurs zero bandwidth overhead and demonstrates privacy protection that is comparable or superior to existing WF defenses.

In this paper, HyWF was configured under specific conditions where the client is multi-homed through two ASes and then passes through a Tor bridge that supports multipath, as illustrated in Fig. 2b. Despite the randomness in the number of sent packets and network path selection, correlations in split traces were observed. Specifically, correlations in their packet counts per time slot, intervals between consecutive packets, and packet sizes were noted.

Given that packets are transmitted independently across two networks, detecting correlations between split traces allows correlated split traces to be smoothly merged, reconstructing the entire trace as if it were collected using a single network.

#### 3) CoMPS

CoMPS, which leverages connection migration features offered by diverse network protocols like QUIC, WireGuard,




(a) TrafficSliver (b) HyWF, CoMPS

**FIGURE 2.** Traffic splitting defenses: TrafficSliver, HyWF, and CoMPS.

**TABLE 1.** The parameters of TrafficSliver, HyWF, and CoMPS.

| Defense | Parameters | |
|---|---|---|
| TrafficSliver | Splitting Strategy | BWR |
| | $n$ | [10, 40], [50, 70] |
| | $w = (w_1, w_2)$ | (0.5, 0.5), (0.3, 0.7) |
| HyWF | $n_{cons}$ | Geometric ($p = \frac{1}{20}$) |
| | $p$ | 0.5 |
| CoMPS | Spitting Strategy | WR |
| | Probability Distribution | Dirichlet |

**TABLE 2.** Various TrafficSliver configurations with each denoted as TS1, TS2, TS3, and TS4.

| | $n$:[10, 40] | $n$:[50, 70] |
|---|---|---|
| $w$:(0.5, 0.5) | TS3 | TS1 |
| $w$:(0.3, 0.7) | TS4 | TS2 |

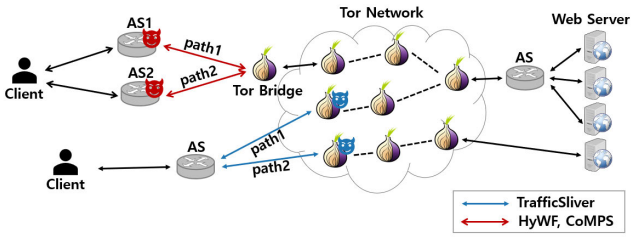



**FIGURE 3.** Threat model: correlation attack on split traces.

and Mosh, was introduced by Wang et al. [18]. Traffic is effectively split by dynamically switching the network addresses for packet transmission, thereby mitigating WF attacks that monitor traffic through one of the networks. In contrast to defenses like TrafficSliver and HyWF that necessitate additional implementations and are confined to specific network protocols like Tor, CoMPS is notable for its ease of deployment and can be readily applied to any network system supporting connection migration.

For path switching, a consistent splitting scheduler was proposed. This scheduler employs various schemes, including round-robin, uniform random, and weighted random, to select the path for packet transmission. Additionally, a context-dependent splitting scheduler was introduced, where switching occurs only in response to specific network events such as handshaking.

The consistent splitting scheme was primarily focused on in their evaluation, with experiments conducted that encompassed all three path selection methods. These methods included round-robin, which alternates paths for packet transmission; uniform random, which uniformly selects paths at random; and weighted random, where probabilities are drawn from a Dirichlet distribution and assigned to paths for each connection. The utilization of a varying number of paths, ranging from two to five, was investigated.

Following previous literature [18], our approach was evaluated against CoMPS with a weighted random path scheduler while maintaining a fixed number of paths. Furthermore, although CoMPS was not originally designed exclusively for the Tor network, its vulnerabilities against correlation attacks were re-evaluated within the context of Tor, as shown in Fig. 2b.

Despite the random selection of multipath, a remarkably consistent pattern in the packet count within fixed short-time intervals, inter-packet delays, and packet sizes for connections directed towards the same destination was observed.

Even though packets traverse the network via multiple paths, each represented by heterogeneous network protocols, the identification of correlations between split connections enables the successful reconstruction of the full trace. This reconstruction mimics the process of collecting traces over a single network protocol, essentially consolidating the information as if transmitted along a single path. Consequently, this approach significantly contributes to high WF accuracy.

In Table 1, we present the selected parameter values for TrafficSliver, HyWF, and CoMPS. Additionally, we investigated four different configurations for TrafficSliver (Table 2) to evaluate their impact on defense performance against DeepCoAST.

## III. DEEPCOAST DETAILS

In this section, the adversary model, the dataset used to evaluate DeepCoAST, its architecture, and the tuning process are presented.

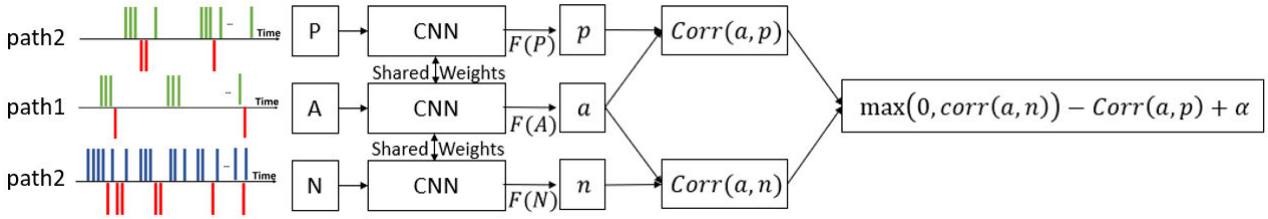



**FIGURE 4.** DeepCoAST's feature extractor.

### A. THREAT MODEL

It is assumed that the adversary is a *global* network-level adversary, similar to those commonly utilized in prior Tor flow correlation research [9], [10], [11]. Additionally, it is posited that users initiate connections to the server through *multiple network paths* designed to split the traffic. The term "multiple network paths" takes on different indications depending on the defense mechanisms employed.

For TrafficSliver, it represents multiple entry nodes in the Tor network. For HyWF, it denotes multiple TCP connections originating from a multi-homed client. For CoMPS, it indicates multiple split connections to various IP addresses and ports, achieved through multiple tunneling protocols using proxies and VPNs. In this paper, it is assumed that the adversary monitors two network paths, denoted as "path1" and "path2," as shown in Fig. 3.

AS-level adversaries can monitor *all* split traces by observing multiple entry guards or controlling ASes. For example, nationwide censorship might intercept all network paths split through multiple guard nodes, multihoming, or connection migration.

Additionally, this attack can be executed using a somewhat less powerful network-level model capable of BGP hijacking. This technique allows the adversary to compel target users to route their traffic through a selected pool of ISPs or ASes, facilitating the monitoring of their split connections despite existing defenses. The overall attack model is illustrated in Fig. 3.

DeepCoAST, the proposed attack, is designed to detect correlations between traffic split through multiple network paths. Once correlated traces are identified, they are merged by ordering Tor cells based on their arrival timestamps, thus reconstructing the full trace. It is important to note that throughout the paper, split trace refers to the sequence of Tor cells rather than packets, with approximately 512 bytes of data encapsulated in each cell. This is because the dataset used was constructed by capturing Tor cell traces. The dataset will be detailed in Section III-B.

### B. DATASET

Defended traces were generated using the BigEnough dataset collected by Matthews et al. [29], which consists of 95 websites, each with 200 Tor cell trace instances. While three browser security levels (standard, safe, and safest) are offered by the dataset, instances collected under the standard security level were specifically utilized for this paper.

As outlined in Table 2 in Section II-E, various configurations for TrafficSliver were further explored, focusing on different values of $n$ and $w$. Here, $n$ represents the number of Tor cells transmitted by a guard node, randomly chosen within specified ranges, and $w$ denotes the path selection probability, also randomly selected within specified ranges.

The original implementations provided by the authors of TrafficSliver,[1] HyWF,[2] and a re-implemented version of CoMPS[3] by Beckerle et al. [30] were utilized to extract defended traces.

Since each defense method significantly reduced the number of cells in split traces, a filtering step was applied to remove very short traces containing fewer than 10 cells. After this filtering process, the dataset consisted of 18,986 pairs for TrafficSliver, 19,000 pairs for HyWF, and 18,780 pairs for CoMPS.

### C. MODEL ARCHITECTURE

As sketched in Fig.4, the feature extraction model of DeepCoAST, trained using a CNN-based architecture, is similar to two Feature Embedding Networks (FENs) of Deep CoFFEA [10] jointly trained using triple loss. Specifically, each FEN in DeepCoAST is composed of four convolutional blocks, with each block consisting of two 1-D convolutional layers followed by a max pooling layer. The ELU activation function was used for the first block, while ReLU was applied to the subsequent blocks. Oh et al. demonstrated that FENs neither required pair-wise correlated or uncorrelated flow pair generations, as DeepCorr [11] did, nor pair-wise similarity score computations for all ingress and egress trace pairs, as RAPTOR [9] conducted. This indicates that FENs achieve less computational complexity than other correlation metrics. We will justify this later in this section.

In addition, DeepCoFFEA is equipped with an evaluator utilizing $k$-Nearest Neighbors ($k$-NN) with cosine similarity metrics. Trace amplification is employed by DeepCoFFEA by evenly dividing the trace into short-time intervals (windows), with each window being evaluated separately. The aggregated results from each window are then used in an

[1] https://github.com/TrafficSliver/splitting_simulator
[2] https://github.com/sebhenri/HyWF
[3] https://github.com/m-bec/Splitting-Hairs-and-Network-Traces/blob/main/simulators/comps-sim.py

ensemble to determine whether the Tor and exit traces belong to the same connection. This approach has been proven to significantly reduce false positives, as agreement from many windows is required.

Two key modifications were made to adapt DeepCoFFEA for correlating split traces. First, considering that split traces through multiple network paths are homogeneous (e.g., in TrafficSliver, they are Tor traffic, and for HyWF and CoMPS, they are encrypted traffic like HTTPS going through a Tor bridge), three models in the feature extractor $F$ were established and trained jointly while sharing weights. This results in a single feature embedding network, or in other words, a unified embedding model similar to the one used in FaceNet [31].

Second, the amplification step of dividing the full trace into windows was excluded. This decision was made based on the observation that many windows in split traces contained only zero cells after dividing each split trace into multiple windows.

According to Fig. 4, $F$ comprises three submodels: anchor, positive, and negative. These submodels are trained jointly to maximize the correlation between embedding pairs from the anchor and positive models while minimizing the correlation between embedding pairs from the anchor and negative models.

For example, in TrafficSliver with two circuits, the traffic of one circuit is fed to the anchor, the traffic of the other circuit on the same connection is fed to the positive, and the traffic of either circuit but on the other connection is fed to the negative.

This objective is captured in the triplet loss function (1).

$$\max(0, \mathsf{Corr}(F(a), F(n)) - \mathsf{Corr}(F(a), F(p)) + \alpha) \quad (1)$$

In this expression, Corr denotes the correlation metric, specifically the cosine similarity chosen for the model. Here, 'a' refers to the input to the anchor model, 'p' corresponds to the input to the positive model, and 'n' represents the input to the negative model.

In the correlation analysis for $n$ correlated pairs, $(t_i, p_j)$, where $1 \leq i, j \leq n$, the triplet $(a, p, n)$ was constructed, with $a = t_1$ and $p = p_1$.

To select an effective $n$ from $p_j$ where $j \neq 1$, the boundary $\alpha$ was empirically set to be farther, but not significantly farther than $p_1$. This hyperparameter $\alpha$ is empirically chosen and serves as a boundary when selecting negative samples for training. Specifically, the negative sample is chosen between the positive sample and $\alpha$. This way, $F$ only requires $n$ triplets for its training to learn the correlation between $n$ correlated pairs.

This approach contrasts with prior methods like RAPTOR [9] and DeepCorr [11], which suffer from quadratic complexity ($n^2$ comparisons) in identifying correlations, as pair-wise feature extractions or correlation score computations are required. This quadratic complexity is a primary reason why triplet loss was chosen to train the feature extractor model of DeepCoAST.

The architecture of the three submodels is built upon the DF [4] architecture, which is one of the most effective traffic fingerprinting models. However, the hyperparameters within the DF model were fine-tuned to enhance its performance, as discussed in Section III-E.

### D. EVALUATION METHODOLOGY

After the feature extractor was trained based on three submodels using the triplet loss, feature embedding vectors were generated using the testing set. Then, pairwise cosine similarity scores between the feature embeddings of split trace pairs were calculated by DeepCoAST, and the top $k$ embeddings were selected.

For instance, as illustrated in Fig. 3, pairwise cosine similarity scores against each trace $p_j$ in path2 were computed for each trace $t_i$ in path1, where $1 \leq i, j \leq n$ and $n$ is the total number of correlated pairs. Here, $i = j$ indicates a correlated pair.

Subsequently, the top $k$ $p_j$'s with the highest scores were selected for each $t_i$. If $p_j$ is present in this top list, it is classified by DeepCoAST as correlated; otherwise, it is classified as uncorrelated. In this context, $k$ serves as the threshold impacting DeepCoAST's performance, and it was varied from 1 to 95, as demonstrated in Section V.

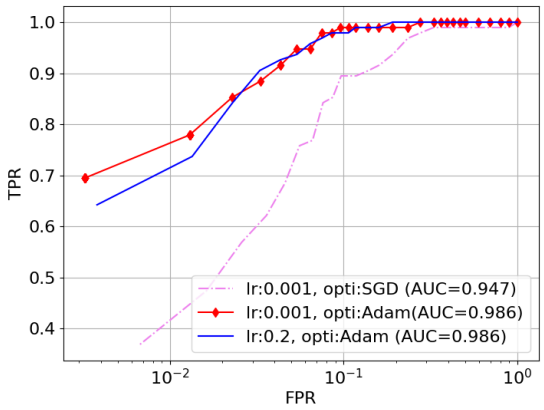



**FIGURE 5.** ROC curves by varying the optimizer and learning rate. Note that the x-axis is a log scale and we used the 1-D TAM features.

### E. HYPERPARAMETER TUNING

To attain superior performance for DeepCoAST, model tuning was conducted against TrafficSliver with the TS1 setting. The search space and the selected parameters are outlined in Table 3. Various search spaces for parameters were explored, including the correlation metric (i.e., Corr) employed in the triplet loss, the boundary $\alpha$ for selecting negative samples, and the number of batches and epochs.

According to Table 3, 128 batches and 300 epochs were used as DeepCoAST demonstrated comparable performance across all batch sizes while requiring less running time with 128 batches. Additionally, improved performance up to 300 epochs was exhibited; however, beyond this point, no further improvement in performance was observed.

**TABLE 3.** Chosen hyper-parameters and search spaces used in the hyper-parameter optimization.

| Param | Chosen Param | Search Space |
|---|---|---|
| Correlation Metric | Cosine similarity | Cosine similarity, Euclidean distance |
| $\alpha$ | $10^{-2}$ | $[10^{-2}...10^{-3}]$ |
| Batch | 128 | [16,32,64,128] |
| Epoch | 300 | [150,300,500,800,1500,2500,3500] |
| Learning Rate | $10^{-3}$ | $[10^{-3}...10^{-4}]$ |
| Optimizer | Adam | Adam, SGD |
| Pooling Layers | Max | Max, Average |
| Activation Function | 1ELU+3RELU | ELU, RELU |
| Number of Blocks | 4 | [4, 5] |
| Number of Filters | | |
| Block1 [Conv1, Conv2] | [32, 32] | [8 ... 64] |
| Block2 [Conv3, Conv4] | [64, 64] | [32 ... 128] |
| Block3 [Conv5, Conv6] | [128, 128] | [64 ... 256] |
| Block4 [Conv7, Conv8] | [256, 256] | [128 ... 512] |
| Dropout | 0.1 | [0.05, 0.1, 0.3, 0.5] |

**Algorithm 1** DeepCoAST Training Algorithm

**Input:** Anchor traces, $A$, positive traces, $P$, negative traces, $N$, a margin, $\alpha$.
**Output:** Network parameters, $\theta$, of the feature extractor model.

**Step 1:** Initialize the shared model, $f$, which has DF architecture.
**Step 2:** Feed $A$, $P$, and $N$ into $f$ to obtain embedding vectors:
$E_A = f(A)$
$E_P = f(P)$
$E_N = f(N)$
**Step 3:** Compute the cosine similarity scores:
$Corr_{AP} = \frac{E_A \cdot E_P}{\|E_A\| \|E_P\|}$
$Corr_{AN} = \frac{E_A \cdot E_N}{\|E_A\| \|E_N\|}$

**Step 4:** Train the triplet model with inputs, $[E_A, E_P, E_N]$
**while** *not converge* **do**
  Compute the triplet loss as follows:
    $L = \max(0, Corr_{AN} - Corr_{AP} + \alpha)$
  Backpropagate the loss $L$.
  Update the triplet network parameters $\theta$.

**Algorithm 2** DeepCoAST Evaluation Algorithm

**Input:** Testing traces on path1, $T$, testing traces on path2, $P$, a threshold, $k$, a feature extractor model, $f$.
**Output:** A true positive count, TP, a false positive count, FP, a true negative count, TN, a false negative count, FN.

**Step 1:** Load the feature extractor model, $f_\theta$.
**Step 2:** Compute the feature embedding vectors using $f_\theta$:
$E_T = f_\theta(T)$
$E_P = f_\theta(P)$
**Step 3:** Compute the pairwise cosine similarity score matrix:
$Corr_{TP} = \frac{E_T \cdot E_P}{\|E_T\| \|E_P\|}$
**Step 4:** Compute TP, TN, FP, FN:
Initialize TP = 0, TN = 0, FP = 0, FN = 0.
Sort the cosine similarity score matrix in an ascending order.
**for** $i = 0; len(E_T) - 1; i++$ **do**
  **for** $j = 0; len(E_P) - 1; j++$ **do**
    **if** $i == j$ *and* $j < k$ **then**
      TP = TP + 1
    **else if** $i \neq j$ *and* $j < k$ **then**
      TN = TN + 1
    **else if** $i == j$ *and* $j \geq k$ **then**
      FN = FN + 1
    **else if** $i \neq j$ *and* $j \geq k$ **then**
      FP = FP + 1
**return** TP, TN, FP, FN

Furthermore, based on the observations in Fig. 5, the Adaptive Moment Estimation (Adam) optimizer with a learning rate of $10^{-3}$ was chosen.

Even though the parameter optimization resulted in the same architecture as the DF model consisting of two convolutional layers per block, known as a CNN block, different but more effective parameter settings for DeepCoAST were identified.

During the hyperparameter optimization process for activation functions, the highest AUC for DeepCoAST was achieved by using the ELU activation function for the first block and the ReLU activation function for the subsequent blocks. Additionally, it was observed that increasing the number of filters in the 1-D convolutional layers enhances the performance of DeepCoAST. The number of filters was set to 32, 64, 128, and 256, respectively, each representing the number of filters in the two convolutional layers of the $i$-th CNN block, where $1 \leq i \leq 4$. Furthermore, a consistent dropout rate of 0.1 was maintained across all blocks.

**TABLE 4. AUC of DeepCoAST when evaluating against TrafficSliver (TS1) by varying *s*.**

| $s$ | 1 | 10 | 100 | 1000 |
|---|---|---|---|---|
| ICD | 0.98 | 0.98 | 0.98 | 0.98 |
| ICDS | 0.93 | 0.98 | 0.99 | 0.98 |

### F. TRAINING & TESTING POLICIES

#### 1) TRAINING PROCEDURE

The training procedure, as outlined in Algorithm 1, is conducted in three steps. The goal is to train the feature extractor model of DeepCoAST to compute feature embedding vectors that are cosine similar for correlated split traces on the same Tor connection.

##### a: STEP 1 - MODEL PREPARATION

The shared model, $f$, based on the DF architecture, is initialized. This model will be used to generate feature embedding vectors for the anchor (A) traces on path1, and the positive (P) and negative (N) traces on path2. Note that positive traces are correlated with the anchor traces as they are split traces on the same connections, while negative traces are not correlated with the anchor traces as they are split traces on different connections.

##### b: STEP 2 - TRAINING EMBEDDING VECTOR GENERATION

The feature embedding vectors for the training set are computed using $f$. $E_A$, $E_P$, and $E_N$ denote the feature embedding vectors of anchor traces, positive traces, and negative traces, respectively.

##### c: STEP 3 - CORRELATION SCORE COMPUTATION

The cosine similarity scores between $E_A$ and $E_P$ and between $E_A$ and $E_N$ are computed.

##### d: STEP 4 - TRAINING WITH A TRIPLET LOSS FUNCTION

Finally, the feature extractor model, $f$, is trained with $E_A$, $E_P$, and $E_N$, using the triplet loss function that maximizes the cosine similarities between $E_A$ and $E_P$ while minimizing the cosine similarities between $E_A$ and $E_N$.

#### 2) TESTING PROCEDURE

The testing procedure, as outlined in Algorithm 2, is conducted in four steps. The goal is to correlate testing split traces on path1 and path2 to determine whether they are from the same Tor connection.

##### a: STEP 1 - MODEL LOADING

The feature extractor model, $f_\theta$, trained in the training procedure is loaded.

##### b: STEP 2 - TESTING EMBEDDING VECTOR GENERATION

The feature embedding vectors for the testing set are computed using $f_\theta$. $E_T$ and $E_P$ denote the feature embedding vectors of traces on path1 and traces on path2, respectively.

##### c: STEP 3 - PAIRWISE CORRELATION SCORE COMPUTATION

The pairwise cosine similarity scores between $E_T$ and $E_P$ are computed.

##### d: STEP 4 - TOP K EMBEDDING SELECTION AND COMPARISON

For the $i$-th vector in $E_T$, $E_{T\,i}$, the pairwise cosine similarity scores are computed against each vector of $E_P$, $E_{P_j}$, where $i = j$ indicates a correlated pair, $1 \leq i, j \leq n$, and $n$ is the total number of correlated pairs.

Then, for each $E_{T\,i}$, the top $k$ vectors with higher cosine similarity scores are selected from $E_P$. If $E_{P_j}$ is present in this top list, it is determined that $E_{T\,i}$ and $E_{P_j}$ are correlated; otherwise, decide it as uncorrelated.

When $i = j$, if $E_{T\,i}$ and $E_{P_j}$ are decided as correlated, it is a TP; otherwise, it is an FN.

When $i \neq j$, if $E_{T\,i}$ and $E_{P_j}$ are decided as correlated, it is an FP; otherwise, it is a TN.

## IV. FEATURE ANALYSIS

In this section, various features employed in earlier WF studies [2], [3], [4], [5], [10], [20], [22] are explored, and an extension of a recent WF feature known as TAM [20] is proposed. This extension aims to optimize TAM as an input vector, enhancing its effectiveness in DeepCoAST.

Let $l$ denote a trace comprising a series of Tor cells $c_i$, where $1 \leq i \leq n$ and $n$ represents the total number of Tor cells in $l$. For each Tor cell $c_i$, the cell direction denoted as $d_i$, cell timing (i.e., arrival time) information denoted as $t_i$, and cell size information denoted as $s_i$ are extracted. Therefore, $l = [(d_1, t_1, s_1), (d_2, t_2, s_2), \ldots, (d_n, t_n, s_n)]$.

### A. CELL DIRECTION

The cell direction, denoted as "+1" when cells are transmitted from the Tor client to the server and "−1" when cells are transmitted from the server to the Tor client, is acknowledged as a valuable feature in WF studies. Consequently, the cell directional sequence is represented as $[d_1, d_2, \ldots, d_n]$.

### B. TIK-TOK

A feature representation was introduced by Rahman et al. [22] by encoding both cell timing and direction information. This was achieved by multiplying each cell's timestamp by its direction. The resulting feature vector is represented as $[d_1 \cdot t_1, d_2 \cdot t_2, \ldots, d_n \cdot t_n]$, and this enhancement significantly improved the performance of DF models, leading to the implementation of Tik-Tok.

### C. INTER CELL DELAY (ICD)

In addition to packet arrival time, the computation of inter-packet delay (IPD), representing the time gap between two consecutive packets, provides informative features and has been widely adopted in WF research [3], [32]. Consequently, the sequence of inter-cell delay (ICD), which indicates the interval between two consecutive cells with a

modification involving multiplication by the cell direction, was evaluated. The resulting feature vector is denoted as $[d_1 \cdot 0, d_2 \cdot (t_2 - t_1), \ldots, d_n \cdot (t_n - t_{n-1})]$. Note that 0 is placed at the beginning, as the first cell should not incur any delays.

Additionally, the values of ICDs were scaled by a factor of $s$, where $s = 1$, 10, 100, and 1,000. This re-scaling might be beneficial in effectively handling small ICDs, such as those below 0.0001, as such values frequently appeared in the ICD sequences. However, as illustrated in Table 4, DeepCoAST demonstrated comparable performance across various values of $s$.

### D. INTER-CELL DELAY WITH SIZE (ICDS)

Inter-cell delay with size (ICDS) was also considered. In DeepCoFFEA [10], IPDs and packet sizes were linearly concatenated in their feature representation, denoted as $[[0, (t_2 - t_1), \ldots, (t_n - t_{n-1})]||[s_1, s_2, \ldots, s_n]]$. IPDs were rescaled by a factor of 1,000 to align their scale with the packet size scale. Following this approach, various scaling factors $s$ from 1 to 1,000 were explored. As shown in Table 4, DeepCoAST demonstrated marginally superior performance with $s = 100$ against TrafficSliver.

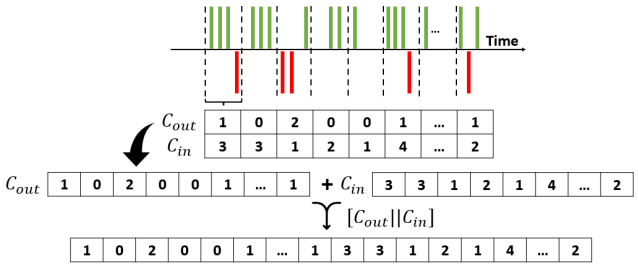



**FIGURE 6.** 1-D TAM feature.

### E. 1-DIMENSIONAL TAM

A feature matrix representing the packet count within each window, called TAM, was introduced by Shen et al. [20]. These windows are short time intervals into which the trace is evenly divided. The feature representation is separately computed for incoming (i.e., $-1$) and outgoing (i.e., $+1$) packet directions, and the results are stacked into a 2-D matrix.

Let $l_{in}$ denote $l_{in} = \{(d_i, t_i, s_i) | (d_i, t_i, s_i) \in l \text{ and } d_i = -1\}$ and $l_{out}$ denote $l_{out} = \{(d_j, t_j, s_j) | (d_j, t_j, s_j) \in l \text{ and } d_j = +1\}$. Note that $i + j = n$. The $i$-th window is denoted as $w_i$ and the packet count in $w_i$ is denoted as $c_i$. For $w_i$, the sequence of packet counts $C_{in}$ for $l_{in}$ and $C_{out}$ for $l_{out}$ is extracted. Then, the original TAM is represented as $[C_{out}; C_{in}]$.

To align with the DF architecture [4], based on 1-D convolutional layers and recognized as one of the most effective traffic fingerprinting models, the 1-D TAM feature was devised by linearly concatenating $C_{out}$ and $C_{in}$, resulting in $[C_{out}||C_{in}]$, as shown in Fig. 6.

To identify more effective features against each of the three defenses, DeepCoAST was evaluated using these features in Section V.

## V. EXPERIMENT DETAILS

In this section, DeepCoAST is assessed as a correlator for detecting correlations between split traces created by three defenses: TrafficSliver, HyWF, and CoMPS. Details on the experimental settings, evaluation metrics, and the performance of DeepCoAST are provided.

### A. SETUP

A single NVIDIA RTX A6000 GPU with 48GB memory was utilized for both training and evaluating DeepCoAST.

For all experiments, the testing set comprised one instance per destination (i.e., website). This ensures that all testing connections are directed to unique destinations, consistent with the setup used in the DeepCoFFEA experiments [10]. Utilizing the BigEnough dataset, which consists of 95 websites, one instance per website was randomly selected, resulting in 95 pairs for the testing set. The remaining pairs were then used for the training set.

### B. EVALUATION METRICS

The performance of DeepCoAST is evaluated using multiple metrics such as True Positive Rate (TPR), False Positive Rate (FPR), Receiver Operating Characteristic (ROC) Curve, F1-Score, Accuracy, and Specificity. These metrics provide a comprehensive assessment of the model's effectiveness and reliability in detecting correlations between split traces.

The following evaluation metrics are used to show the performance of DeepCoAST:

- **True Positive Rate (TPR, Sensitivity)**: The fraction of correctly identified correlated pairs over all correlated pairs. The performance of DeepCoAST in detecting the presence of a specific correlation is measured by this metric.

$$TPR(Sensitivity) = \frac{TP}{TP + FN} \tag{2}$$

  where TP (True Positive) is the number of correctly identified correlated pairs, and FN (False Negative) is the number of correlated pairs that were incorrectly identified as uncorrelated.
- **False Positive Rate (FPR)**: The fraction of incorrectly identified uncorrelated pairs over all uncorrelated pairs. It is important to note that the number of uncorrelated pairs is $n^2 - 1$ for $n$ correlated pairs. A low value for this metric is indicative of the effectiveness of the correlator, considering the pairwise comparisons involved.

$$FPR = \frac{FP}{FP + TN} \tag{3}$$

  where FP (False Positive) is the number of uncorrelated pairs that were incorrectly identified as correlated, and TN (True Negative) is the number of correctly identified uncorrelated pairs.
- **Receiver Operating Characteristic (ROC) Curve**: Instead of fixing the confidence threshold at a specific value, varying the threshold allows for a comprehensive

assessment of model performance. As discussed in Section III-D, $k$ was varied, representing the number of path2 traces with the highest cosine similarity scores. Using these different values of $k$, both TPRs and FPRs were visualized and the AUC was subsequently calculated. A larger AUC signifies the greater effectiveness of the model.

- **F1-Score**: The F1-Score acts as the harmonic mean of precision and recall, providing a balance between the two. Given the known tradeoff between precision and recall when varying the thresholds, the F1-Score is beneficial as it reflects both precision and recall. The score is computed as follows:

$$F1\text{-}Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{4}$$

where,

$$Precision = \frac{TP}{TP + FP}$$

and

$$Recall = \frac{TP}{TP + FN}$$

- **Accuracy**: The accuracy is computed as the fraction of correctly determined pairs, whether correlated or uncorrelated. It provides an overall measure of the model's performance. The formula for accuracy is as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

- **Specificity**: The specificity is computed as the fraction of correctly identified, uncorrelated pairs among all uncorrelated pairs. It measures the ability of the model to correctly identify uncorrelated pairs. The formula for specificity is as follows:

$$Specificity = 1 - FPR \tag{6}$$

## VI. RESULTS AND DISCUSSIONS

In this section, we discuss the performance of DeepCoAST by exploring diverse features, defense parameters, and various evaluation metrics to demonstrate their impact. Furthermore, we summarize our key findings and discuss potential future work to address the identified limitations.

### A. DEEPCOAST PERFORMANCE

DeepCoAST was evaluated by exploring various features, defenses, and configurations (Table 2), particularly in TrafficSliver (TS).

#### 1) FEATURES

As discussed in Section IV, various popular WF features were applied to DeepCoAST to assess their effectiveness in correlation detection. This exploration involved five feature sets across all three defenses.

**TABLE 5. AUC of DeepCoAST by varying defenses, features, and TS configurations.**

| Feature | TS1 | TS2 | TS3 | TS4 | HyWF | CoMPS |
|---|---|---|---|---|---|---|
| Direction | 0.87 | 0.88 | 0.90 | 0.91 | 0.94 | 0.93 |
| Tik-Tok | 0.98 | 0.98 | 0.98 | 0.98 | 0.99 | 0.98 |
| 1-D TAM | 0.98 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| ICD($s$=1,000) | 0.98 | 0.98 | 0.99 | 0.98 | 0.99 | 0.99 |
| ICDS($s$=1,000) | 0.98 | 0.99 | 0.99 | 0.98 | 0.99 | 0.98 |

As observed in Fig. 7, different feature rankings were resulted by the defenses, although 1-D TAM and ICD features consistently occupied top positions. Notably, directional feature vectors did not exhibit a significant level of correlated traffic patterns, in contrast to traditional WF models where they played a central role, as observed in well-known WF models [4], [5].

In addition, the utilization of only cosine similarity without a feature extractor yielded poor performance across all defenses, consistent with the findings in DeepCoFFEA [10]. This underscores the significance of the deep learning-based feature extractor in detecting correlations between split traces across multiple network paths.

#### 2) DEFENSES

The performance of DeepCoAST across three defenses was further compared. As illustrated in Fig. 8, the correlation between split traces was successfully identified by DeepCoAST when utilizing 1-D TAM features under all three defenses. This result raises significant concerns, as split traces displayed a notable level of correlation, achieving an AUC of over 0.98 at best.

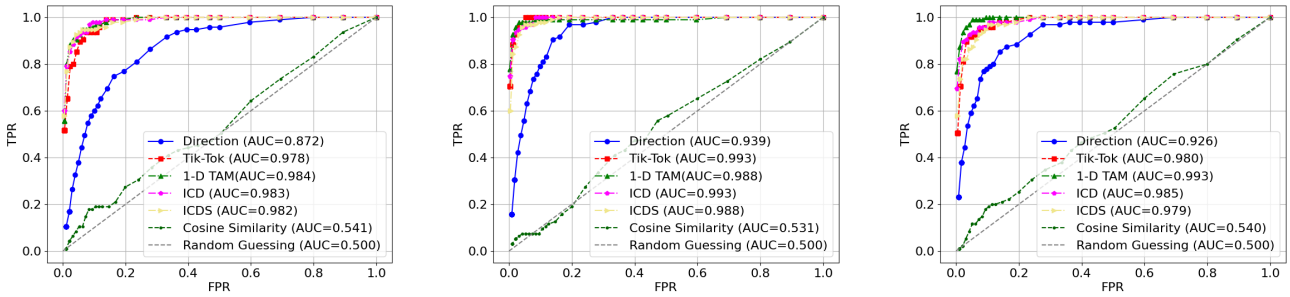
Additional features for all defenses were explored, as presented in Table 5. Overall, superior performance in detecting correlations was demonstrated by DeepCoAST, particularly with Tik-Tok, 1-D TAM, and ICD features, across all three defenses.

#### 3) TS PARAMETERS

To investigate the impact of parameters $n$ (the number of cells to be sent) and $w$ (the probability to select the path) in TrafficSliver, four parameter combinations were explored, as shown in Table 2. The performance of DeepCoAST is presented in Table 5, and Fig. 9 indicates that these parameters marginally impacted DeepCoAST, especially with more powerful features such as Tik-Tok, 1-D TAM, and ICD. The extension of this analysis to cover more parameter sets is left for future work.

#### 4) OTHER EVALUATION METRICS

When $k$ increases from 1 to 95, both TPR and FPR increase, as shown in Fig.7. However, Fig.7 does not clearly show the performance of DeepCoAST by precision scores. To address this, $k$ is fixed at 5, and the F1-scores are computed by considering both TPRs and precision scores. According to Table 6, even at over 90% TPR, F1-scores were




(a) TrafficSliver (b) HyWF (c) CoMPS

**FIGURE 7.** ROC curve of DeepCoAST by varying features and defenses. Note that $s = 100$ for ICD and ICDS.

**TABLE 6.** The accuracy, F1-Score, TPR(Sensitivity), FPR, specificity of DeepCoAST by varying defenses and features (when, threshold $k = 5$).

| Defense | Feature | Accuracy | F1-Score | TPR(Sensitivity) | FPR | Specificity |
|---|---|---|---|---|---|---|
| TS1 | Direction | 94.45 | 0.13 | 37.89 | 4.95 | 95.05 |
| | Tik-Tok | 95.48 | 0.28 | 85.26 | 4.41 | 95.59 |
| | 1-D TAM | 95.59 | 0.30 | 90.53 | 4.36 | 90.53 |
| | ICD($s$=1,000) | 95.55 | 0.29 | 88.42 | 4.38 | 95.62 |
| | ICDS($s$=1,000) | 94.97 | 0.20 | 61.05 | 4.67 | 95.33 |
| HyWF | Direction | 94.84 | 0.19 | 55.79 | 4.75 | 95.25 |
| | Tik-Tok | 95.77 | 0.33 | 98.95 | 4.27 | 95.73 |
| | 1-D TAM | 95.75 | 0.33 | 97.89 | 4.28 | 95.72 |
| | ICD($s$=1,000) | 95.58 | 0.30 | 90.53 | 4.37 | 95.63 |
| | ICDS($s$=1,000) | 94.50 | 0.13 | 38.95 | 4.90 | 95.10 |
| CoMPS | Direction | 94.93 | 0.20 | 58.95 | 4.69 | 95.31 |
| | Tik-Tok | 95.61 | 0.31 | 91.58 | 4.34 | 95.66 |
| | 1-D TAM | 95.72 | 0.32 | 96.84 | 4.29 | 95.71 |
| | ICD($s$=1,000) | 95.59 | 0.30 | 90.53 | 4.36 | 95.64 |
| | ICDS($s$=1,000) | 95.04 | 0.21 | 64.21 | 4.64 | 95.36 |

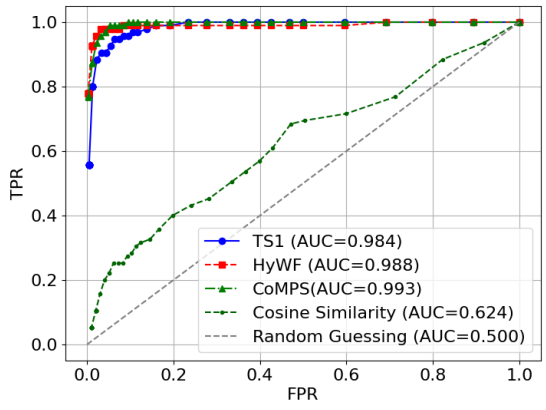



**FIGURE 8.** Comparison of TrafficSliver (TS1), HyWF, and CoMPS using the 1-D TAM features.

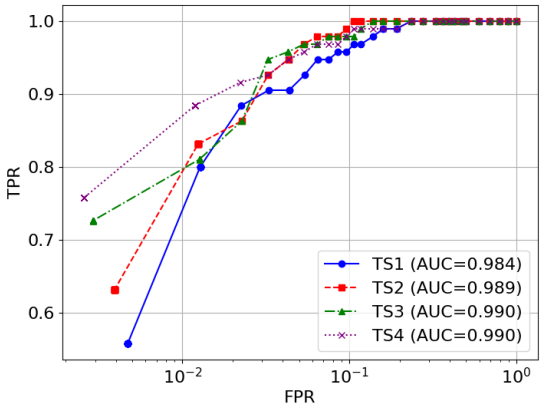



**FIGURE 9.** Comparison of TS1, TS2, TS3, and TS4 using the 1-D TAM features. Note that the x-axis is a log scale.

around 30%, indicating that the number of false positives should be reduced further through potential improvements. However, as presented in Table 6, the FPRs were under 5%, and specificity scores were over 95%, indicating that uncorrelated pairs were accurately identified in all cases. Further improvements to achieve sufficiently high precision are left for future work.

### B. SUMMARY OF FINDINGS

We summarize our findings through experiments in Section VI.

- DeepCoAST demonstrated robust performance against three well-known traffic splitting defenses: TrafficSliver, HyWF, and CoMPS.

- Both traditional WF features and their variants, along with the modified feature embedding models of DeepCoFFEA, make DeepCoAST advantageous as a split traffic correlation detector.
- The performance of DeepCoAST remains effective even when varying parameters of the three defenses.

With these findings, our correlation analysis of split traffic by WF defenses urges Tor researchers to improve traffic splitting defenses to prevent leaking the correlation between split traces of the same connection.

### C. FUTURE WORK DISCUSSIONS

Even though DeepCoAST shows potential as a split traffic correlator, this paper includes some limitations. In this section, we discuss potential future work aimed at enhancing the robustness of DeepCoAST and conducting more realistic evaluations.

#### 1) LARGER TESTING DATASET

The construction of the testing set, which includes unique destinations based on the BigEnough dataset, resulted in some artificial settings. In the real world, many destinations, including some overlapping destinations, would appear in multiple Tor connections.

As the potential of DeepCoAST as a correlator to correlate 95 split traces heading to unique destinations has been successfully demonstrated, the current experimental scenarios will be extended to evaluate DeepCoAST against a much larger dataset with overlapping destinations. This will result in more realistic experimental settings.

#### 2) END-TO-END FLOW CORRELATION

Even though the effectiveness of DeepCoAST in detecting the correlation between split traces has been demonstrated, the extent to which this approach helps identify website traces under splitting defense techniques is questionable.

For the end-to-end flow correlation attack, the attack can be carried out in two stages. First, once the correlated traces are determined by DeepCoAST, they are merged to recover a single path trace. Second, the WF models are evaluated using these recovered traces.

With a higher correlation rate demonstrated by AUC, DeepCoAST would result in lower false predictions while identifying websites. The second stage of the attack also allows for a fair comparison to other WF attacks against splitting defenses. Although this paper focuses on the first stage of the attack, the second stage is left as future work.

#### 3) EVALUATION USING MORE NETWORK PATHS

The experiments primarily focus on evaluating DeepCoAST using split traces collected from two network paths. However, considering that TrafficSliver and CoMPS are configurable with more than three paths, exploring this additional setting would be a reasonable next step. As a possible extension to detect the correlation between $n$ split traces, where $n \geq 3$, DeepCoAST is utilized sequentially.

For example, the testing set comprising $a_i$, $b_j$, and $c_k$, where $1 \leq i, j, k \leq n$ and $n$ is the total number of connections, with each collected from path1, path2, and path3, respectively, is assumed. DeepCoAST first detects the correlation between $a_i$ and $b_j$, sorts out correlated pairs, and merges each pair (determined as correlated) as a single path trace. Then, DeepCoAST computes the correlation between these recovered traces and $c_k$, consequently deriving the final correlated triplets.

Further elaboration on this approach, including how to compute WF metrics to properly measure the performance of this style of attack, could be an interesting area for future work.

#### 4) DEEPCOAST WITH AMPLIFICATION

As a significant loss in the number of packets in split traces was experienced, the amplification setting in the DeepCoFFEA implementation was canceled. This decision was made as the amplification further divided the split traces into multiple small windows, leading to an increase in the amount of padding. However, previous literature [10] has demonstrated the effectiveness of amplification in considerably reducing the number of false positives, as they are determined with enough votes as correlated from multiple windows.

To incorporate amplification into DeepCoAST, longer connections will be captured. Specifically, by monitoring Tor connections for a longer duration (e.g., more than one minute, which was a flow duration used in DeepCoFFEA [10]), further investigation into how much amplification enhances the correlation capability will be conducted in future work.

## VII. CONCLUSION

In this paper, the effectiveness of end-to-end flow correlation attacks against two split traces generated by traffic splitting defenses using a new split trace correlation model, DeepCoAST, was investigated.

While traffic splitting defense techniques have been recognized as robust WF defenses, DeepCoAST can lead to the merging of split traces into a single trace, resulting in high WF accuracies as shown in previous WF literature. A thorough feature analysis based on WF features was conducted to suggest more effective correlated features for DeepCoAST, and DeepCoFFEA was carefully tailored to build a robust architecture of the feature extractor model against split traces. Across three defenses, multiple features, and various TrafficSliver configurations, DeepCoAST demonstrated effective performance with an AUC of 0.98.

Based on the experimental results, an urgent need for further research on additional defense mechanisms to counter DeepCoAST attacks exists, given the considerable amount of correlated traffic pattern leakage. Additionally, further studies on DeepCoAST evaluation against more network path

settings and the enhancement of DeepCoAST remain future work.

## REFERENCES

[1] A. Mani, T. Wilson-Brown, R. Jansen, A. Johnson, and M. Sherr, "Understanding tor usage with privacy-preserving measurement," in *Proc. Internet Meas. Conf.*, Oct. 2018.

[2] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proc. 23rd USENIX Secur. Symp. (USENIX Secur.)*, 2014, pp. 143–157.

[3] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *Proc. 25th USENIX Secur. Symp. (USENIX Security)*. Austin, TX, USA: USENIX Association, Aug. 2016, pp. 1187–1203.

[4] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Oct. 2018, pp. 1928–1943.

[5] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," 2017, *arXiv:1708.06376*.

[6] S. Eun Oh, S. Sunkam, and N. Hopper, "P-FP: Extraction, classification, and prediction of website fingerprints with deep learning," 2017, *arXiv:1711.03656*.

[7] P. Sirinam, N. Mathews, M. S. Rahman, and M. Wright, "Triplet fingerprinting: More practical and portable website fingerprinting with N-shot learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Nov. 2019, pp. 1131–1148.

[8] S. E. Oh, N. Mathews, M. S. Rahman, M. Wright, and N. Hopper, "GANDaLF: GAN for data-limited fingerprinting," *Proc. Privacy Enhancing Technol.*, vol. 2021, no. 2, pp. 305–322, Apr. 2021.

[9] Y. Sun, A. Edmundson, L. Vanbever, O. Li, J. Rexford, M. Chiang, and P. Mittal, "RAPTOR: Routing attacks on privacy in Tor," in *Proc. 24th USENIX Secur. Symp. (USENIX Secur.)*, 2015, pp. 271–286.

[10] S. E. Oh, T. Yang, N. Mathews, J. K. Holland, M. S. Rahman, N. Hopper, and M. Wright, "DeepCoFFEA: Improved flow correlation attacks on tor via metric learning and amplification," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2022, pp. 1915–1932.

[11] M. Nasr, A. Bahramali, and A. Houmansadr, "DeepCorr: Strong flow correlation attacks on tor using deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Oct. 2018, pp. 1962–1976.

[12] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, I still see you: Why efficient traffic analysis countermeasures fail," in *Proc. IEEE Symp. Secur. Privacy*, May 2012, pp. 332–346.

[13] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, "A systematic approach to developing and evaluating website fingerprinting defenses," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2014, pp. 227–238.

[14] J. Gong and T. Wang, "Zero-delay lightweight defenses against website fingerprinting," in *Proc. USENIX Secur. Symp.*, 2020.

[15] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an efficient website fingerprinting defense," in *Proc. 21st Eur. Symp. Res. Comput. Secur. (ESORICS)*, Heraklion, Greece, Sep. 2016, pp. 27–46.

[16] W. De la Cadena, A. Mitseva, J. Hiller, J. Pennekamp, S. Reuter, J. Filter, T. Engel, K. Wehrle, and A. Panchenko, "TrafficSliver: Fighting website fingerprinting attacks with traffic splitting," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Oct. 2020, pp. 1971–1985.

[17] S. Henri, G. Garcia-Aviles, P. Serrano, A. Banchs, and P. Thiran, "Protecting against website fingerprinting with multihoming," *Proc. Privacy Enhancing Technol.*, vol. 2020, no. 2, pp. 89–110, Apr. 2020.

[18] M. Wang, A. Kulshrestha, L. Wang, and P. Mittal, "Leveraging strategic connection migration-powered traffic splitting for privacy," *Proc. Privacy Enhancing Technol.*, vol. 2022, no. 3, pp. 498–515, Jul. 2022.

[19] *Tor Design Proposals: Overcoming tor's Bottlenecks With Traffic Splitting*. [Online]. Available: https://spec.torproject.org/proposals/329-traffic-splitting.html

[20] M. Shen, J. Zhang, L. Zhu, K. Xu, and X. Du, "Subverting website fingerprinting defenses with robust traffic representation," in *Proc. 32nd USENIX Secur. Symp.*, 2023, pp. 607–624.

[21] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel, "Website fingerprinting at internet scale," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2016.

[22] M. S. Rahman, P. Sirinam, N. Mathews, K. G. Gangadhara, and M. Wright, "Tik-tok: The utility of packet timing in website fingerprinting attacks," *Proc. Privacy Enhancing Technol.*, vol. 2020, no. 3, pp. 5–24, Jul. 2020.

[23] S. Bhat, D. Lu, A. Kwon, and S. Devadas, "Var-CNN: A data-efficient website fingerprinting attack based on deep learning," *Proc. Privacy Enhancing Technol.*, vol. 2019, no. 4, pp. 292–310, Jul. 2019.

[24] X. Cai, R. Nithyanand, and R. Johnson, "CS-BuFLO: A congestion sensitive website fingerprinting defense," in *Proc. 13th Workshop Privacy Electron. Soc.*, Nov. 2014.

[25] T. Wang and I. Goldberg, "Walkie-talkie: An efficient defense against passive website fingerprinting attacks," in *Proc. 26th USENIX Secur. Symp. (SEC)*, Vancouver, BC, Canada, Aug. 2017, pp. 1375–1390.

[26] J. Asharf, N. Moustafa, H. Khurshid, E. Debie, W. Haider, and A. Wahab, "A review of intrusion detection systems using machine and deep learning in Internet of Things: Challenges, solutions and future directions," *Electronics*, vol. 9, no. 7, p. 1177, Jul. 2020.

[27] R. M. S. Priya P. K. R. Maddikunta, M. Parimala, S. Koppu, T. R. Gadekallu, C. L. Chowdhary, and M. Alazab, "An effective feature engineering for DNN using hybrid PCA-GWO for intrusion detection in IoMT architecture," *Comput. Commun.*, vol. 160, pp. 139–149, Jul. 2020.

[28] M. K. Nallakaruppan, S. R. K. Somayaji, S. Fuladi, F. Benedetto, S. K. Ulaganathan, and G. Yenduri, "Enhancing security of host-based intrusion detection systems for the Internet of Things," *IEEE Access*, vol. 12, pp. 31788–31797, 2024.

[29] N. Mathews, J. K. Holland, S. E. Oh, M. S. Rahman, N. Hopper, and M. Wright, "SoK: A critical evaluation of efficient website fingerprinting defenses," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2023, pp. 969–986.

[30] M. Beckerle, J. Magnusson, and T. Pulls, "Splitting hairs and network traces: Improved attacks against traffic splitting as a website fingerprinting defense," in *Proc. 21st Workshop Privacy Electron. Soc.*, vol. 42, New York, NY, USA, Nov. 2022, pp. 15–27.

[31] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 815–823.

[32] G. D. Bissias, M. Liberatore, D. Jensen, and B. N. Levine, "Privacy vulnerabilities in encrypted HTTP streams," in *Proc. Int. Workshop Privacy Enhancing Technol.* Cavtat, Croatia. Berlin, Germany: Springer, 2005, pp. 1–11.

**GOUN KIM** received the B.S. degree in computer science and engineering from Ewha Womans University, Seoul, South Korea, in 2021, where she is currently pursuing the M.S. degree in AI convergence. She has been actively involved as a Graduate Researcher with the AI Security (AISec) Laboratory, where she conducts research on traffic analysis for network security, autonomous vehicle security, and user authentication.

**HYEONJEONG KWAK** is currently pursuing the B.S. degree in computer science and engineering with Ewha Womans University. She is actively involved in an internship for undergraduate students with the AISec Laboratory, Ewha Womans University, where she is conducting research on network security, specifically utilizing deep learning techniques. Her primary interests include machine learning and networking.

**SUJIN KIM** is currently pursuing the B.S. degree in artificial intelligence and computer science and engineering (minor) with Ewha Womans University. She is also an Undergraduate Research Student with the AISec Laboratory, Ewha Womans University.

**YOUHEE PARK** is currently pursuing the dual B.S. degree in food science and engineering and in computer science with Ewha Womans University. She has been actively engaged in an internship with the AISec Laboratory for a year, where she serves as a Researcher focusing on network security. Her work involves utilizing deep learning and machine learning skills to contribute to the field.

**JIHYEUN PARK** is currently pursuing the B.S. degree in computer science with engineering with Ewha Womans University. In 2022, she completed a one-year internship with the AISec Laboratory. Her current research interest includes the application of AI models to various domains.

**SE EUN OH** (Member, IEEE) received the M.S. degree in computer science from the University of Illinois at Urbana–Champaign, and the Ph.D. degree in computer science and engineering from the University of Minnesota. She is an Assistant Professor with the Computer Science and Engineering Department and a Director of the AISec Laboratory, Ewha Womans University. She has published several research papers in the domain of anonymity and privacy within prestigious security and privacy venues, including the IEEE Symposium on Security and Privacy (S&P) and the Proceedings on Privacy Enhancing Technologies Symposium (PoPETS). Additionally, she has served as a reviewer for reputable security journals, such as IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING and PoPETS.