STAP: Leveraging State-Transition Adversarial Perturbations for Asymmetric Website Fingerprinting Defenses

STAP: Leveraging State-Transition Adversarial Perturbations for Asymmetric Website Fingerprinting Defenses

Jia-Nan Huang, *Student Member, IEEE*, Weiwei Liu, *Member, IEEE*, Guangjie Liu, *Member, IEEE*, Bo Gao, Fengyuan Nie, Marco Mellia, *Fellow, IEEE*

***Abstract*—Web services, as the most ubiquitous form of online services, have consistently attracted research attention due to privacy concerns. Although VPNs and anonymous communication methods can partially protect users' online privacy, advancements in website fingerprinting (WF) attacks still exploit the spatio-temporal characteristics of web resource transmission to identify web services. The challenge lies in defending against WF attacks efficiently, with limited bandwidth costs. Server-side WF defenses, deployed on web servers, can achieve end-to-end obfuscation across both clients and servers. However, existing defenses often consume significant bandwidth and require additional removal operations on the client side. Given the growing use of QUIC with HTTP/3 and the need for robust privacy protections, this paper introduces an asymmetric server-side WF defense scheme using State-Transition Adversarial Perturbations (STAP). STAP introduces the concept of latent resource-state transitions, which represent hidden patterns in resource transmission. Utilizing perturbation models containing these transitions, STAP subtly alters traffic through packet padding and insertion, with inherent transport layer encryption enhancing the concealment. STAP can operate independently, removing the necessity for user involvement. Experimental results demonstrate that STAP outperforms other schemes, achieving reductions in True Positive Rate (TPR) by up to 22% and reductions in bandwidth overhead by up to 30%.**

***Index Terms*—Privacy protection, website fingerprinting defense, HTTP/3, adversarial perturbations, traffic analysis.**

I. Introduction

With the ubiquity of web services in both the Internet and the Internet of Things (IoT) [1], online privacy protection has become a critical concern that consistently attracts research attention. The widespread adoption of HTTP/3 has introduced the end-to-end encryption protocol QUIC, which enhances privacy by encrypting certificates and minimizing plaintext exposure during transmission. Coupled with this, many users employ privacy-enhancing tools (PETs) like VPNs and anonymous communication methods, which use secure and private protocols to encrypt network traffic. However, despite these advancements, website fingerprinting (WF) attacks remain a significant challenge. These attacks can be executed through traffic analysis systems deployed at various points in the network infrastructure, including Internet service providers (ISPs), autonomous systems (AS), and even compromised PET nodes [2]. These attacks exploit the spatio-temporal characteristics of web resource transmissions to infer users' activities by employing classifiers to distinguish between traffic patterns associated with different web services [3]. Among the various techniques used, machine learning-based methods have emerged as the primary analysis tools [4].

To defend against WF attacks, WF defenses aim to modify the spatio-temporal characteristics of resource transmissions through packet manipulations [5]. Simple scheme involves regularizing all packets to the same length and interval time [6], but this introduces unacceptable bandwidth overhead and degrades web service quality. Researchers have also explored introducing randomness [7] to obscure definitive patterns and adversarial defense schemes [8] to deceive WF attacks. To enable practical implementations of such techniques, Mullvad VPN has developed DAIST functions, which support both regularization and randomization to obscure website resource transmission patterns and counter increasingly sophisticated machine learning-based traffic analysis [9]. Additionally, some schemes adjust browser configurations and resource files to alter the web service behavior, thereby affecting the identification of website browsing traffic [10], [11].

Existing WF defense schemes can be categorized into two main types based on their operational processes: symmetric and asymmetric defenses. Symmetric defenses modify traffic during transmission and restore it before delivery; asymmetric defenses operate independently without requiring restoration, making them more suitable for real-world deployment.

When it comes to implementation, client-side defenses are typically deployed on the user's device, offering adaptability and flexibility. For instance, symmetric defense schemes are often integrated within PETs [12], executing client-side defenses through collaboration with PET nodes. However, these client-side defenses fail to protect traffic between PET nodes and web servers, leaving it vulnerable to flow correlation attacks [13]. Additionally, defenses can place an extra burden

This work was supported by the National Natural Science Foundation of China (Grants No. U2436601, U21B2003, 62072250, 61602247) and the National Key R&D Program of China (Grants No. 2021QY0700). *(Corresponding author: Weiwei Liu.)*

Jia-Nan Huang, Weiwei Liu, Bo Gao, Fengyuan Nie are with the School of Automation, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: jiananwong@njust.edu.cn, lwwnjust@njust.edu.cn, njustgb565@163.com, niefengyuan@njust.edu.cn).

Guangjie Liu is with the School of Electronics and Information Engineering, Nanjing University of Information Science and Technology, Nanjing 210044, China and also with Key Laboratory of Intelligent Support Technology for Complex Environments, Ministry of Education, Nanjing, China (e-mail: gjieliu@gmail.com).

Marco Mellia is with the Department of Control and Computer Engineering, Politecnico di Torino, 10129 Turin, Italy (e-mail: marco.mellia@polito.it).

on client-side devices, particularly those with limited computational resources, such as IoT devices. On the other hand, server-side defenses, deployed on web servers or their reverse proxies, provide end-to-end obfuscation for both clients and servers. These defenses can be implemented by simply adding a plugin to the server, making the deployment process straightforward. The combination of server-side implementation and asymmetric defenses emerges as a promising scheme, as it offers privacy protection without requiring user involvement and cooperation.

Nevertheless, existing server-side schemes exhibit inherent limitations on their asymmetric implementation. For example, in [11], resource files such as images and texts are modified, which not only requires elevated permissions on the server side but also raises privacy concerns stemming from resource personalization, and may compromise the integrity of the original files. Similarly, the approach proposed in [14] involves padding SSL-based web applications, which necessitates client-side cooperation for the removal of padding. This dependency introduces potential vulnerabilities, particularly if adversaries are aware of the padding strategy [15]. These challenges underscore the necessity for a defense mechanism that maintains the integrity of resource transmission during traffic manipulation and eliminates reliance on client-side modifications. QUIC, as the underlying protocol of HTTP/3, emerges as an ideal candidate for implementing packet-level manipulation and achieving end-to-end obfuscation. QUIC offers several advantageous features for this purpose. First, it natively supports payload padding via PADDING frames, which carry no semantic content and are automatically discarded by the receiver, requiring no client-side intervention for packet reconstruction. Second, QUIC's frame-based architecture ensures that the original application-layer payload remains intact and unaffected by the added padding. Moreover, due to its near-complete encryption, QUIC inherently obscures packet-level manipulations from passive observers, thereby enhancing the stealth and effectiveness of the defense. As a result, these characteristics make QUIC particularly suitable for supporting packet manipulation strategies that aim to achieve robust, asymmetric, and end-to-end traffic obfuscation.

Beyond addressing implementation challenges, existing schemes often consume significant bandwidth without thoroughly analyzing the trade-offs between increased bandwidth costs and enhanced defense effectiveness. Many of them offer shallow, cost-oblivious solutions that fail to address the need for optimizing both cost-efficiency and defense effectiveness simultaneously. In another, server-side WF defenses should be sufficiently robust to withstand adversarial WF attacks. In this scenario, attackers actively visit and collect traffic from websites implementing WF defenses, train classifiers using the collected samples, and infer users' activities based on the trained models.

In this paper, we introduce STAP, a novel asymmetric adversarial WF defense framework that leverages **S**tate-**T**ransition **A**dversarial **P**erturbations to enhance end-to-end obfuscation between web servers and users. STAP achieves this by manipulating traffic packets on the server side through strategic padding and insertion operations. The foundation of STAP is its multi-layered perturbation models, constructed through iterative adversarial training. This training capitalizes on latent resource-state transitions extracted from the packet lengths of resource transmission and adapts the morphing website selection distribution through the discriminator's top-N predicted probability. During operation, each perturbation model selects a morphing website target, generating a sequence of latent resource-states that the traffic must align with. This alignment through various perturbation models effectively introduces perturbations, thereby obscuring the web traffic pattern and mitigating adversarial WF attacks. Considering STAP deployment, we leverage QUIC, a novel protocol that supports both encryption and padding. Specifically, QUIC's built-in encryption features ensure that modifications remain concealed, while its padding frame design facilitates server-side WF defenses without necessitating any specific removal operations from the client application.

The main contributions of this study can be outlined as follows:

- We introduce STAP, an innovative asymmetric WF defense framework that integrates packet manipulations to counter adversarial WF attacks through State-Transition Adversarial Perturbations. Operating on the server side, STAP ensures end-to-end obfuscation between clients and web servers, eliminating the need for additional modifications on the client side. The inherent encryption capabilities and frame design of QUIC naturally facilitate these manipulations, making STAP a practical solution in actual deployment.
- We design a novel packet manipulation method for WF defenses based on multi-layered perturbation models crafted through adversarial training. Each model selects a morphing website target and generates sequences of the target website's latent resource-states. These states encapsulate the resource-state transitions of different websites during packet transmission, which are used to align actual service packets, thereby introducing perturbations.
- To evaluate the effectiveness of STAP, we employ state-of-the-art traffic analysis tools and compare STAP against advanced WF defense schemes. For a direct comparison, we introduce three configurations of STAP. Across three datasets, STAP demonstrates an average of 10%-20% lower bandwidth overhead and achieves an average reduction of about 10%-20% in True Positive Rate (TPR) compared to the four WF defense schemes evaluated.

The remainder of this paper is structured as follows: Section II provides an overview of related work in WF attacks techniques and WF defense schemes. Section III discusses the adversary model and introduces the framework of STAP. Section IV describes the construction of perturbation model within STAP. Section V conducts experimental analysis and discusses results. Finally, Section VI concludes this paper.

II. Related Work

In this section, we present the evolution of WF attack techniques, which have progressed from rule matching-based methods to shallow learning-based approaches, and more

recently, to deep learning-based techniques. We also provide an overview of WF defenses, including regularization-based, randomization-based, adversarial schemes, and others.

A. WF Attack Techniques

Within the rule matching methods, deep packet inspection (DPI) is a traditional technique that analyzes the contents of captured traffic packets using prior knowledge. Kopek et al. [16] utilize signatures extracted from packet payloads and integrate them with Snort for analysis, while Husák et al. [17] use a set of matching rules, including user-agent and handshake information, for analyzing HTTPS protocol.

Although DPI can respond rapidly, its accuracy heavily relies on the completeness of the knowledge base. Given the need for dynamic updates of this knowledge and challenges related to encryption protocols, current research increasingly favors the use of machine learning models. Chang et al. [18] utilize both message type and length sequences to construct Markov models for analyzing encrypted traffic, while Anderson et al. [19] consider packet length and time distributions, analyzing them with logistic regression models. FlowPrint [20] uses a smaller set of features, including IP, port, timestamp, and TLS certificate, to classify traffic flows via clustering. However, these shallow learning-based methods heavily rely on feature selection, necessitating extensive expertise.

Current research is increasingly focused on deep learning, with sequential representations providing a new perspective for enhancing traffic flow representation [21]. FS-Net [22] adopts a bidirectional gated recurrent unit (GRU) to extract features from encrypted traffic's packet length sequences. To improve memory capabilities for sequence data, NeuTic [23] integrates a sequence self-attention mechanism with multi-kernel convolution to identify long-range dependencies. Additionally, ET-BERT [24] applies the bidirectional encoder representations from Transformers (BERT) model for pre-training, aiming at classifying encrypted traffic through contextualized datagram representations. However, ET-BERT's dependence on certain plaintext fields restricts its suitability for complex analysis. To overcome this challenge, AppSniffer [25] introduces a stacked ensemble model that combines Light Gradient Boosting Machine with FastAI models for a two-step approach to advanced analysis.

B. WF Defense Schemes

With the increasing sophistication of WF attacks leveraging spatio-temporal characteristics of website browsing, various WF defense schemes have been proposed to implement traffic obfuscation through different strategies.

Regularization-based defense schemes focus on mitigating traffic characteristic leaks. BuFLO [26], an initial effective approach, regularizes all packets' length and interval time to eliminate feature leakage. Enhancements like Tamaraw [6] has a smaller packet length and increased interval times in traffic to reduce overhead, and CS-BuFLO [27] introduces dynamic transmission rate adjustments. Despite their effectiveness, these methods significantly increase bandwidth consumption and latency due to their stringent measures. RegulaTor [28], building on these concepts, modifies traffic into decaying packet surges, striving for equilibrium while still incurring unsustainable bandwidth overhead. In another, Prism [29] alters the distribution of packet lengths and arrival times towards less common patterns, complicating traffic pattern identification.

Randomization-based defenses leverage randomness to ensure that traffic flows within the same website do not exhibit identical patterns. WTF-PAD [7] conceals distinctive timing gaps in traffic flows by sampling from predetermined distributions and inserting dummy packets during these gaps when no packets are being sent. FRONT [30] applies the Rayleigh distribution to send dummy packets at the start of traffic flows, capitalizing on the principle that the initial segment of the flows is rich in distinguishing features. BLANKET [31] utilizes random noise to generate perturbations via packet padding, insertion, and delaying as a defense strategy against non-adversarial WF attacks.

Adversarial defenses are a new direction in crafting WF defense schemes, capitalizing on the vulnerability of deep neural networks (DNN)-based classifiers to small, crafted perturbations on the inputs. Mockingbird [8] utilizes the Euclidean distance between packet bursts for packet insertion, defending against DNN-based classifiers but requiring the knowledge of whole traffic flows. Dolos [32] analyzes collected traffic flows, concentrating on the Euclidean distance of feature space parameters from DNN-based classifiers, and then generates positions for inserting packets. Minipatch [33] enhances practicality by using an optimization algorithm to insert packets and utilizing the DNN-based classifier probability result for defense effectiveness evaluation.

While the aforementioned defenses achieve obfuscation through fine-grained packet manipulation, they require intermediary nodes for traffic restoration because obfuscated packets deviate from standard structures. To address these limitations and achieve end-to-end privacy protection, application layer WF defenses have been proposed. These defenses follow the syntax and semantics of application messages to asymmetrically modify resource transmission patterns. For instance, Walkie-Talkie [10] modifies web browsers to operate in half-duplex mode, sequencing responses to disrupt the temporal patterns of web resources. TrafficSliver [34] distributes resource requests across multiple flows, enabling parallel transmission to obfuscate website fingerprints. ALPaCA [11] embeds additional bytes into resource files, altering resource transmission patterns to achieve obfuscation. However, application layer obfuscations are limited to coarse-grained manipulations of application messages, making it challenging to balance bandwidth overhead and obfuscation effectiveness against widely utilized packet-level WF attacks.

In contrast to these existing WF defenses, STAP is integrated into the QUIC protocol, enabling fine-grained packet manipulations and providing end-to-end privacy protection. Furthermore, STAP leverages QUIC's unique padding frames to achieve asymmetric defenses without requiring client-side collaboration. This approach effectively obfuscates website fingerprints while maintaining compatibility with standard protocols.

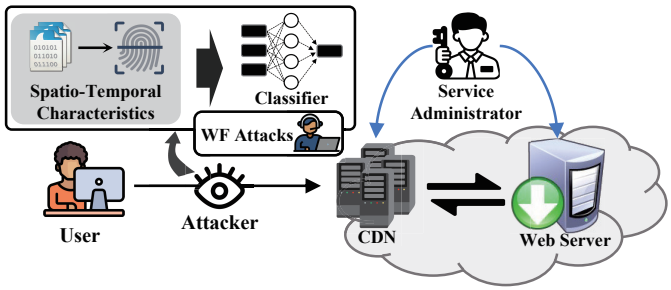

Fig. 1. The adversary model of WF attacks.

III. System Design

In this section, we delineate the adversary model and present the problem formulation aimed at countering adversarial WF attacks. Subsequently, we depict the proposed framework, which implements the defense scheme against WF attacks via packet manipulations.

A. Adversary Model

We illustrate the adversary model in Figure 1. To reduce pressure on the actual web servers, service administrators commonly rely on content delivery networks (CDNs). These enhance the user experience and naturally protect the web servers by acting as intermediaries. By hosting multiple websites that share one or more IP addresses, CDNs inherently anonymize server identifiers, e.g., its IP addresses, making it challenging for WF attacks to directly utilize this information to distinguish between different web services [35]. Despite these measures, WF attacks can exploit the spatio-temporal characteristics of web resource transmission to infer online activities.

Under this adversary model, the attacker is characterized as a passive observer who can monitor all network traffic to and from the CDN nodes without the ability to decrypt or modify it. The primary goal of the attacker is to identify network flows corresponding to specific websites served by the CDN nodes. Due to the fact that network flows are prone to varying degrees of congestion from routing nodes, which disrupts packet interval time-based features, attackers prefer to utilize packet lengths for more reliable spatio-temporal analysis. By utilizing traffic analysis tools, the attacker trains classifiers on a dataset of traffic samples to accurately identify the specific website associated with each traffic flow. Consequently, the widespread use of CDN nodes does not impede the attacker's capability to perform WF attacks. Here we assume that users access only one website at a time, simplifying the attacker's task by eliminating the need to distinguish between multiple websites accessed simultaneously.

To defend against WF attacks, the adversarial training-based WF defense should follow the paradigm described below:

$$\begin{aligned} &\hat{t}^* = \arg\max_{\hat{t}} \mathbb{E}_{(\hat{t},w)\in(\hat{T},Y)} Loss(\psi^*(\hat{t}), w) \\ &\text{s.t.} \quad \psi^* = \arg\min_{\psi} \mathbb{E}_{(t,w)\in(T,Y)} Loss(\psi(t), w) \\ &\qquad K = \mathcal{F}(T) \\ &\qquad \hat{T} = \{\hat{t} \mid \hat{t} = \mathcal{M}(t, K), \forall t \in T\} \\ &\qquad O(\hat{t}) < \delta, \forall \hat{t} \in \hat{T} \end{aligned} \tag{1}$$

In this formulation, $t \in T$ and $\hat{t} \in \hat{T}$ represent the undefended and obfuscated traffic flows for the website $w \in Y$, respectively. T and $\hat{T}$ denote the undefended and obfuscated datasets, and Y is the corresponding set of labels. The problem definition focuses on training the optimal discriminator $\psi^*(\cdot)$ and subsequently optimizing the obfuscated traffic $\hat{t}$ in relation to $\psi^*(\cdot)$. The loss distance for the discriminator is calculated using $Loss(\cdot,\cdot)$ with the ultimate goal of obtaining the optimal obfuscated traffic $\hat{t}^*$ that maximizes the loss of the discriminator.

To achieve this goal, the perturbation model K is first extracted from the dataset T using the extraction function $\mathcal{F}(\cdot)$. Then, the undefended traffic t is obfuscated by applying the packet manipulation function $\mathcal{M}(\cdot,\cdot)$ using the perturbation model K, resulting in the obfuscated traffic $\hat{t}$. Finally, to ensure practical applicability, we introduce the bandwidth and latency overhead constraint, where $O(\cdot)$ calculates the overhead, and δ represents the maximum allowable overhead for the obfuscated traffic $\hat{t}$.

B. Overview of STAP

As depicted in Figure 2, STAP is a novel asymmetric adversarial defense scheme designed to provide end-to-end privacy protection against WF attacks. Operating at the server side, STAP manipulates packets during QUIC encapsulation, leveraging QUIC's unique features to eliminate the need for client-side restoration. The defense scheme operates in two stages: perturbation model training and perturbation manipulation. During the first stage, STAP constructs multi-layered perturbation models through adversarial training framework to determine ideal number of perturbation layers. In the manipulation phase, these trained models are integrated into the QUIC protocol stack to guide real-time packet operations, employing packet padding and insertion, that obfuscate observable traffic patterns.

1) Perturbation Model Training: To enhance obfuscation performance and ensure adaptability to various website traffic patterns, STAP employs an iterative adversarial training process to construct its multi-layered perturbation models. Each perturbation model builds upon the obfuscation introduced by the previous layer. During the v-th round, the dataset T^v is used to generate the v-layer perturbation model K^v in the perturbation model construction component. This perturbation model is then evaluated in the self-assessment evaluation component, where K^v applies perturbations to T^v to produce the perturbed dataset T^{v+1}. A benefit-cost analysis is performed to assess the effectiveness of the obfuscation. If further improvement is needed, T^{v+1} is used in the subsequent round to generate perturbation model K^{v+1}. The detailed training process is described in Section IV.

To maximize obfuscation effectiveness, each perturbation model is designed to capture two essential elements: the distribution of morphing websites and the resource transmission patterns associated with these targets. For morphing website distribution, STAP employs adversarial training to leverage the discriminator's output probabilities, which are used to select the most suitable obfuscation targets for each website traffic sample. For resource transmission pattern modeling, STAP

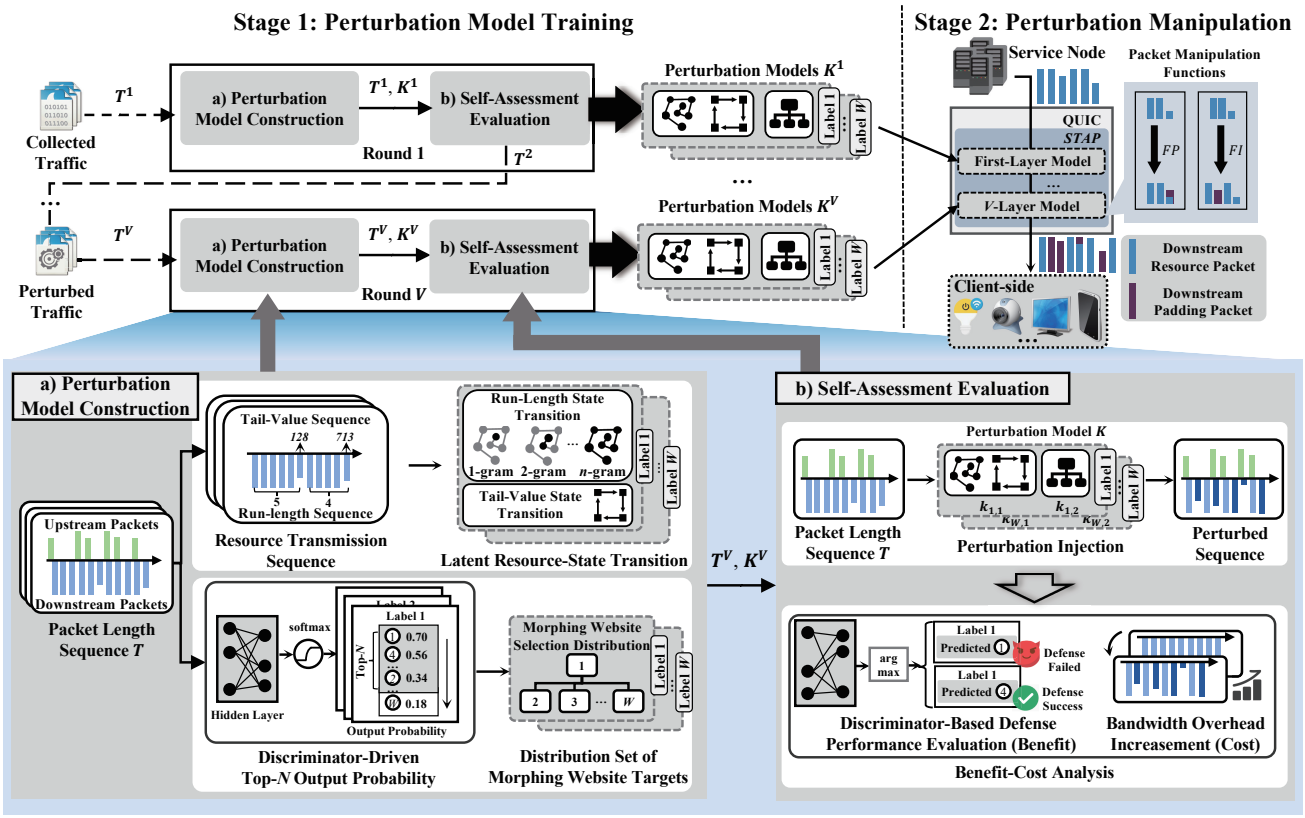

Fig. 2. The overview of STAP.

introduces the concept of latent resource-state transitions, which encapsulate the underlying dynamics of resource transmissions across different websites to capture the interactions and variations in resource transmission behaviors.

These transitions are identified from a distinct pattern observable in web resource transmission that the intermittent presence of packets with payloads smaller than the Maximum Payload Size (MPS)[1]. These non-MPS packets typically appear after a sequence of several MPS packets. This distinctive pattern facilitates the identification of latent resource-states, which capture the underlying dynamics of resource transmissions from the traffic flows. These states are delineated by partitioning the payload length sequence of resource packets, specifically identifying non-MPS packets. Each segment thus corresponds to the latent transmission of a distinct resource or a collective block of resources.

Formally, consider a resource packet payload length sequence comprising I non-MPS packets. The latent resource-state $\pi_{w,i}$ during the transmission of the i-th resource within website w is represented as:

$$\pi_{w,i} = [x_{w,i}, l_{w,i}] \tag{2}$$

Here, $x_{w,i}$ represents the number of MPS packets between the $(i-1)$-th and the i-th non-MPS packets, while $l_{w,i}$ denotes the payload length of the i-th non-MPS packet. This representation captures the characteristics of individual resource transmissions and can be further refined into two sequences: the run-length sequence of MPS packets, denoted by X_w, and the tail-value sequence of non-MPS packets, denoted by L_w. These sequences are expressed as follows:

$$\begin{cases} X_w = [x_{w,1}, x_{w,2}, \cdots, x_{w,I}] \\ L_w = [l_{w,1}, l_{w,2}, \cdots, l_{w,I}] \end{cases} \tag{3}$$

Here, $x_{w,i}$ represents the run-length state, and $l_{w,i}$ denotes the tail-value state of the i-th resource transmission. For example, consider a website w with two latent resource-states: $\pi_{w,1} = [5, 128]$ and $\pi_{w,2} = [4, 713]$. This indicates that the first resource transmission consists of five MPS packets followed by a non-MPS packet of 128 bytes, while the second consists of four MPS packets followed by a non-MPS packet of 713 bytes. The corresponding sequences are $X_w = [5, 4]$ and $L_w = [128, 713]$. These sequences of latent resource-states capture the state transitions in web resource transmissions, enabling the prediction of future resource-states based on current and preceding states.

2) Perturbation Manipulation: Once the training is completed, STAP deploys the trained multi-layered perturbation models on the service node, serving as a pluggable transport layer within QUIC. When resources are requested, they are encapsulated through QUIC and added obfuscation by STAP before sending. STAP adheres to QUIC specifications for

[1]MPS is derived from the maximum packet size constraint imposed by data link layer technologies, such as the Maximum Transmission Unit (MTU) of Ethernet.

packet manipulations and leverages the protocol's padding frames to introduce perturbations, enabling its asymmetric WF defense scheme with end-to-end obfuscation that requires no additional removal operations on the client side. Due to the unpredictable nature of future packets, STAP manipulates packets individually through two packet manipulation functions:

Packet Padding $FP(p_a, \xi)$: Appends QUIC padding frames to packet p_a until its total payload length matches ξ.

Packet Insertion $FI(p_a, \xi)$: Inserts a packet with payload length ξ before the packet p_a, filled solely with QUIC padding frames.

As STAP introduces state-transition adversarial perturbations, the downstream flow needs to be sequentially aligned with the latent resource state generated by each perturbation model through packet manipulation functions. For each packet p_a in the resource packet payload length sequence P^r, STAP employs the manipulation decision algorithm (Algorithm 1) to determine the appropriate perturbations. The algorithm initializes a buffer $\mathcal{B}$ for each p_a, storing the packets to be sent. It then sequentially applies trained V-layered perturbation models, $K^1 - K^V$, to introduce perturbations. Each perturbation model K^v generates a predicted latent resource-state sequence, where each latent resource-state $\pi^v_{w,i}$ depends on previous latent resource-states following the knowledge of resource-state transitions within the respective perturbation model K^v. The predicted latent resource-state sequence is then converted into a predicted payload length sequence P^v.

Then, the algorithm compares the current buffer packet's payload length p_b with the maximum payload size $n_{\max}$ and the predicted payload length $pred$, selecting either FP or FI to align the packet accordingly. The subsequent perturbation model operates on the output of the previous model, using the updated buffer $\mathcal{B}$. This iterative process continues until all V perturbation models have been applied. It is worth noting that the calculation of predicted payload lengths can be pre-generated or performed concurrently with ongoing packet manipulations, ensuring a high level of responsiveness.

Algorithm 1: Manipulation Decision Algorithm

Input: $P^v = \{p^v_1, p^v_2, \dots\}$ - Predicted payload length sequence from the v-layer perturbation model, where $v \in \{1, 2, \dots, V\}$
$P^r = \{p_1, p_2, \dots\}$ - Resource packet payload length sequence
$n_{\max}$ - Maximum payload size for UDP traffic
$FP(p_m, \xi)$ - Packet Padding Function
$FI(p_m, \xi)$ - Packet Insertion Function

Output: Resource packets with perturbations

```
Offset 𝒪 ← [1] * V;
each p_a in P^r;
Buffer ℬ ← [p_a];
for P^v in {P^1, P^2, ..., P^V} do
    BufferNew ℬ_n ← [];
    for i ← 0 to len(ℬ) do
        p_b ← ℬ[i];
        while True do
            pred ← p^v_{𝒪[v]+len(ℬ_n)};
            if pred ≤ n_max and p_b < n_max then
                p_b ← FP(p_b, max(p_b, pred));
            else if pred < n_max and p_b = n_max then
                ℬ_n.append(FI(p_b, pred));
                continue;
            ℬ_n.append(p_b);
            break;
    ℬ ← ℬ_n;
    𝒪[v] ← 𝒪[v] + len(ℬ_n);
Send(ℬ);
```

IV. Perturbation Model Training in STAP

In this section, we present the perturbation model training method utilized by STAP. The training process comprises two components: perturbation model construction and self-assessment evaluation. Perturbation model construction is responsible for constructing perturbation models that capture the knowledge of latent resource-state transitions and morphing website selection distribution for each website. Subsequently, self-assessment evaluation determines the suitable number of perturbation models to ensure that the current multi-layered perturbation models effectively enhance defense performance without introducing excessive overhead.

A. Perturbation Model Construction

In this step, the perturbation model K is constructed for each website w. It consists of two components: $k_{w,1}$ and $k_{w,2}$. The component $k_{w,1}$ captures the knowledge of latent resource-state transitions, which is extracted from the packet payload lengths of resource transmission. Simultaneously, $k_{w,2}$ represents the morphing website selection distribution, derived from the top-N predictions provided by the discriminator.

1) Latent Resource-State Transition: The component contains two types of state transition: run-length and tail-value. To capture the run-length state transition, we apply the n-gram model, commonly used in natural language processing. And each run-length is considered as a word, and sequences of consecutive lengths are considered as sentences. For a given website w, the set of n gram-based models, $\{gram^1_w, gram^2_w, \cdots, gram^n_w\}$, provide the run-length state transition of resource transmission features. In $gram^n_w$, the probability for transitioning to the i-th run-length state $x_{w,i}$, based on the sequence of the preceding $(n-1)$ states, is calculated as:

$$Pr(x_{w,i} \mid x_{w,i-n+1}, \cdots, x_{w,i-1}) = \frac{freq(x_{w,i-n+1}, \cdots, x_{w,i})}{freq(x_{w,i-n+1}, \cdots, x_{w,i-1})} \tag{4}$$

where $freq(\cdot)$ denotes the frequency of occurrences for each run-length state sequence that belongs to website w.

To capture the tail-value state transition, it is essential to recognize the dynamic nature of web resource transmission, which can lead to byte-level variations in identical resources. These variations range from a few bytes to several tens of bytes and significantly influence the payload length of non-MPS packets. We introduce the tail-value state transition matrix $P_w = (p_{i,j})^{M \times N}$ to capture this variability. This matrix

provides a view to anticipate and analyze changes in packet sizes. In P_w, the rows divide the payload lengths of non-MPS packets into M segments, starting from the MPS value. Each column corresponds to the potential lengths of upcoming non-MPS packets. Specifically, each element $p_{m,n}$ indicates the probability of transition from the current non-MPS payload length in the m-th segment to a potential length for the next non-MPS packets in the n-th position.

2) Distribution of Morphing Website Targets: Since classifiers used by attackers commonly employ softmax as the final activation function, our discriminator also uses it to provide output probabilities for classifying traffic flows into various websites. Higher probabilities indicate a greater similarity between the traffic flow and the corresponding websites. Utilizing this information, STAP can select a highly correlated morphing website target to activate the perturbation. This strategy, compared to selecting a random target, enhances defense efficiency and maintains reasonable bandwidth consumption by aligning the traffic with the most similar target.

For any given website w, the morphing targets are determined by examining the top-N predicted probability, excluding the probability of the website being identified as itself. The probability that a traffic flow from website w morphs into the i-th website out of W options is calculated by the following equation:

$$Pr(w_i|w_1, \cdots, w_W) = \frac{n_i^\tau}{\sum_{j=1}^{W} n_j^\tau} \tag{5}$$

Here, n_i represents the discriminator's predicted probability of website w being classified as website w_i. τ is a parameter regulating the smoothness of the distribution. As τ tends to 0, the website probabilities become nearly uniform, while increasing τ towards infinity progressively differentiates between the probability of distinct websites, facilitating a more selective approach to identifying morphing website targets for WF defenses.

B. Self-Assessment Evaluation

In this step, STAP determines whether to end the iterative training process based on the enhanced defense performance and the incremental bandwidth overhead. Specifically, each traffic flow $\{t, w\} \in \{T^v, Y\}$ is obfuscated through perturbation injection using the perturbation model $K^v = \{k_{w,1}, k_{w,2}\}$ in the v-th round. After the obfuscation, the discriminator's accuracy and the bandwidth overhead are measured through benefit-cost analysis.

1) Perturbation Injection: This process involves the assignment of a morphing website target to each traffic flow and modifying its sequence of latent resource-state to align with the generated one. Initially, a weighted-based random sampling method is introduced to determine the order of target websites for each website to morph into. For a flow belonging to website w, $k_{w,2}$ contains the probability of selecting a potential morphing website w_u. These probabilities are then converted into order weights (ϱ_u) using the weighted random sample in [36]:

$$\varrho_u = rand(0,1)^{\frac{1}{w_u}} \tag{6}$$

where $rand(0,1)$ generates a random number between 0 and 1. The potential morphing website targets are then ranked in descending order based on their order weights. The top Φ websites are selected as the morphing targets for the current website, which will morph into these Φ targets sequentially. Once the morphing process for these Φ targets is completed, a new set of morphing targets is regenerated using the same method.

Upon selecting the morphing website target w_u, the knowledge of latent resource-state transition, $k_{w_u,1}$, is applied. We introduce a run-length state prediction method that leverages decisions from multiple n-gram models to update the i-th run-length state. It incorporates weighted integration, enabling each submodel to proportionally influence the overall prediction. The i-th run-length state $x^*_{w,i}$ is calculated as follows:

$$x^*_{w,i} = \arg\max_{x_{w,i}} \sum_{j=1}^{n} q_{j,i} G^j_{w_u}(x_{w,i}) \tag{7}$$

where $G^j_{w_u}(x_{w,i}) = Pr(x_{w,i} \mid x^*_{w,i-j+1}, \cdots, x^*_{w,i-1})$ represents the conditional probability of $x_{w,i}$ in the j-gram model $gram^j_{w_u}$, taking into account the preceding states $\{x^*_{w,i-j+1}, \cdots, x^*_{w,i-1}\}$. These preceding states are also obtained through the same way. The term $q_{j,i}$ indicates the reliability weight of the $G^j_{w_u}(x_{w,i})$, with weights dynamically adjusted based on the $(i-1)$-th run-length state prediction.

$$q_{j,i} = \frac{G^j_{w_u}(x^*_{w,i-1})}{\sum_{k=1}^{n} G^k_{w_u}(x^*_{w,i-1})} \tag{8}$$

The higher the prediction accuracy of an n-gram model, the more it contributes to the decision-making process for the next state.

In the process of modifying the tail-value state l_i of the i-th resource, the transition matrix P_{w_u} guides the selection of the subsequent state l_i based on the previous state l_{i-1}, which corresponds to the m-th segment $\boldsymbol{p_m}$ within P_{w_u}. The probabilities for transitioning from l_{i-1} to potential future states are detailed within vector $\boldsymbol{p_m} = \{p_{m,1}, p_{m,2}, \ldots, p_{m,n}\}$, where each element of $\boldsymbol{p_m}$ represents the likelihood of evolving to a specific tail-value state. To derive the tail-value state for the i-th resource, the inverse cumulative distribution function is employed, leveraging the probabilities encoded in $\boldsymbol{p_m}$.

2) Benefit-Cost Analysis: STAP employs multi-layered perturbation models to enhance defense effectiveness against adversarial WF attacks. While increasing the number of perturbation layers generally improves defense by introducing further modifications to the packets, it also results in higher bandwidth overhead. Importantly, the increase in defense effectiveness due to additional bandwidth does not always follow a linear or exponential growth pattern. Therefore, a benefit-cost analysis is critical to determine the suitable number of perturbation layers V^* for defensive efficacy.

To navigate this trade-off, we use a net benefit metric E_v to assess the combined impact of each perturbation layer during the v-th training round. This metric evaluates the balance between enhanced defense performance, indicated by a decrease in Accuracy (Acc) of the discriminator, and the increase in bandwidth overhead (BWO) compared to the last training round. The formula used is:

$$E_v = \frac{\Delta Acc_v}{\Delta BWO_v} = \frac{Acc_{v-1} - Acc_v}{BWO_v - BWO_{v-1}} \tag{9}$$

In this formulation, ΔAcc_v represents the improvement in defense effectiveness, and ΔBWO_v quantifies the additional bandwidth overhead introduced in each round. The net benefit metric E_v measures the efficiency of defense improvement per unit of bandwidth cost. When $E_v > 1$, each unit of added bandwidth yields a greater-than-proportional gain in defense performance, indicating an efficient trade-off. Conversely, when $E_v \leq 1$, the marginal gain in defense does not justify the bandwidth increase. Therefore, the training process terminates, and the model from the previous round is retained. By capturing the per-round trade-off between accuracy reduction and bandwidth cost, E_v provides a unified performance measure to evaluate defense efficiency and serves as a key criterion for determining the optimal stopping point in STAP's multi-layered perturbation training.

V. Experimental Results and Analysis

In this section, we perform extensive experiments to compare STAP against four advanced WF defense schemes. These comparisons are conducted using two state-of-the-art traffic analysis tools. Furthermore, we validate the performance of STAP and provide a detailed analysis based on publicly available datasets.

A. Experimental Setup

1) Datasets: We utilize three HTTP/3-QUIC website browsing traffic datasets: a single-connection dataset D_{conn}, a multi-connection dataset D_{mconn}, and a browser-connection dataset D_{bconn}. All three datasets are sourced from highly-ranked websites on Alexa that support QUIC [37]. The first dataset D_{conn} consists of network flows collected from 100 websites, each represented by 100 samples, along with 10,000 unmonitored websites. Each flow in this dataset is captured using the QUIC client library to download website resources, with traffic restricted to a single QUIC connection. The second dataset D_{mconn} encompasses network flows that span multiple connections. It includes 12,000 samples from the same 100 websites and an additional 2,000 samples from unmonitored websites. The third dataset D_{bconn} uses the Chromium browser to download web resources. This dataset comprises 20,000 samples from the same 100 websites and an 10,000 unmonitored website samples.

2) Settings: To achieve a comprehensive and meaningful evaluation, the compared defense schemes must demonstrate both effective obfuscation and practical feasibility. Specifically, these schemes should operate on real-time traffic without requiring knowledge of future packets or the ability to modify previously transmitted packets. Based on these criteria, we compare STAP with four advanced WF defense schemes: WTF-PAD [7], Prism [29], FRONT [30], and Minipatch [33], which serve as suitable benchmarks for comparison. Specifically, WTF-PAD sends dummy packets at distinct timing gaps using pre-configured distributions. FRONT inserts dummy packets into the initial traffic segment. Minipatch inserts blocks of dummy packets to bolster defense with minimal overhead. Prism disrupts the traffic pattern by modifying the frequency characteristics of the spatio-temporal features. For fairness, each scheme uses the parameters recommended in the original papers. Additionally, while some of these methods are typically implemented using PET nodes to execute symmetric WF defenses, we do not consider this scenario in our experiments. These compared defense schemes are treated as server-side defenses that manipulate downstream packets to provide end-to-end privacy protection from server to client, which falls within the scope of our asymmetric defense framework.

To evaluate the effectiveness of WF defense schemes, we test it against state-of-the-art traffic analysis tools. We consider two DNN-based classifiers: NeuTic [23] and AppSniffer [25]. NeuTic employs a self-attention mechanism with multi-kernel convolution to capture long-range dependencies on the spatio-temporal characteristics of web resource transmission. On the other hand, AppSniffer utilizes a stacked ensemble model that combines Light Gradient Boosting Machine and FastAI models for traffic identification.

In our experiments, half of the samples from each website are allocated for the construction of perturbation models, with the remainder being obfuscated and evaluated. All the codes run on a platform with an Intel i9-14900KF CPU with 64 GB RAM and an NVIDIA GeForce RTX 4090 GPU with 24 GB memory. The implementation is done in Python 3.8.19 and PyTorch 2.1.2. We make available our source code at https://github.com/Jia-nan-wong/stap.

3) Metric: The bandwidth overhead (BWO) is a key metric for evaluating the effectiveness of WF defense schemes. It is calculated as:

$$BWO = \frac{\sum_{t \in T} \mid \mathcal{M}(t, K) \mid - \mid t \mid}{\sum_{t \in T} \mid t \mid} \tag{10}$$

where $\mid t \mid$ represents the total transmission bytes of traffic flow t, and $\mid \mathcal{M}(t, K) \mid$ is the total transmission bytes of traffic flow obfuscated through perturbation models K.

In addition to overhead metrics, the results from traffic analysis tools are crucial in evaluating defense effectiveness. We utilize six commonly used metrics for WF defense assessment: Accuracy (Acc), F1-score ($F1$), Recall (R), Precision (P), True Positive Rate (TPR), and False Positive Rate (FPR), which are explicitly calculated as follows:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

$$R = \frac{TP}{TP + FN} \tag{12}$$

$$P = \frac{TP}{TP + FP} \tag{13}$$

$$TPR = \frac{TP}{TP + FN} \tag{14}$$

$$FPR = \frac{FP}{TN + FP} \tag{15}$$

$$F1 = \frac{2 \times P \times R}{P + R} \tag{16}$$

where TP, FP, TN, and FN denote the true positive, false positive, true negative, and false negative, respectively. A higher defense performance is characterized by lower Acc, R, P, TPR and $F1$, as well as a higher FPR.

B. The Selection of Parameters

This section examines the defense performance of STAP across different parameters of the n-gram model in the run-length state transition and using various traffic analysis tools as the discriminator within a closed-world environment. In this setting, we assume that user only visit websites from a list monitored by the attacker, who aims to accurately classify the traffic into its correct website. Accuracy is calculated as the proportion of correctly classified samples. While this environment does not reflect real-world conditions, it provides a controlled setting for the assessment of STAP.

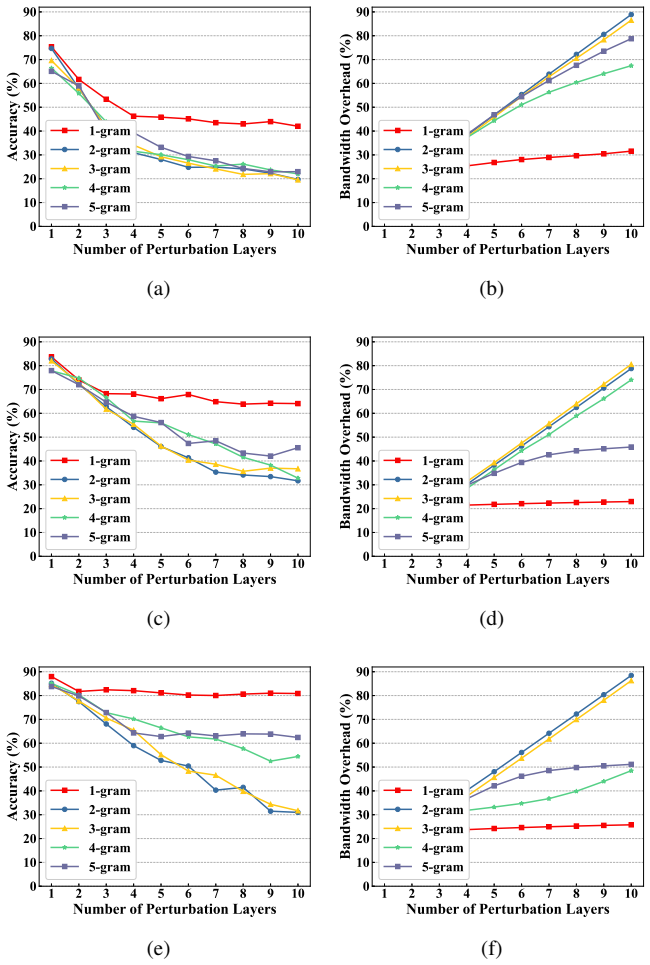

Fig. 3. Classification accuracy and bandwidth overhead with different numbers of perturbation models: (a) Acc on the dataset D_{conn}; (b) BWO on the dataset D_{conn}; (c) Acc on the dataset D_{mconn}; (d) BWO on the dataset D_{mconn}; (e) Acc on the dataset D_{bconn}; (f) BWO on the dataset D_{bconn}.

Figure 3 presents the results obtained through five-fold cross-validation to enhance reliability. The findings indicate that increasing the number of perturbation layers leads to a decline in classification accuracy, accompanied by a rise in bandwidth overhead. The effectiveness of various n-gram models differs depending on the dataset and its intrinsic transmission characteristics. The 1-gram model performs poorly across all datasets, as it relies on random sampling without leveraging contextual information, causing early convergence and limited defense performance.

For the dataset D_{conn}, the 4-gram model achieves comparable defense effectiveness to other models while incurring slightly lower bandwidth overhead. This suggests that the 4-gram model effectively captures the single-connection transmission dynamics of D_{conn}, enabling more accurate pattern prediction by leveraging longer contexts. However, extending context length does not universally enhance performance; in datasets with more complex transmission patterns, longer contexts may reduce model efficiency.

In contrast, the dataset D_{mconn}, characterized by multi-resource transmissions with high variability, benefits more from the 2-gram model. Similarly, the dataset D_{bconn}, which includes both single- and multi-connection resource transmissions, exhibits a level of contextual complexity that is also best addressed by the 2-gram model.

Based on these observations, the 4-gram model is identified as optimal for D_{conn}, while the 2-gram model is better suited for the more complex D_{mconn} and D_{bconn}. Furthermore, as the number of perturbation layers increases, the rate of accuracy decline gradually slows, leading to diminishing returns in defense effectiveness and unnecessary bandwidth overhead. Our benefit-cost analysis identifies $V^* = 4$ as the optimal number of perturbation layers across all three datasets, achieving a well-balanced trade-off between defense performance and bandwidth consumption.

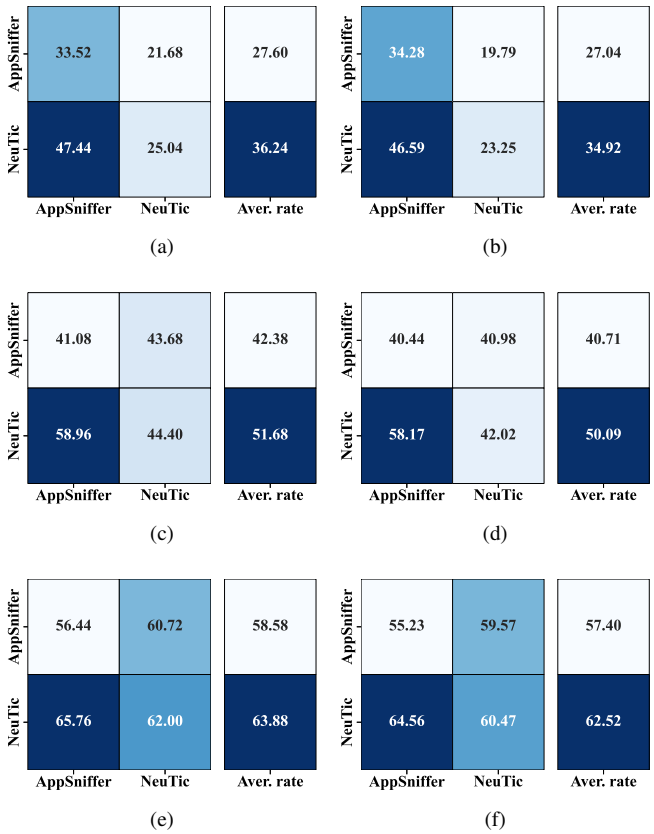

Fig. 4. Confusion matrix of different discriminators: (a) Acc on the dataset D_{conn}; (b) $F1$ on the dataset D_{conn}; (c) Acc on the dataset D_{mconn}; (d) $F1$ on the dataset D_{mconn}; (e) Acc on the dataset D_{bconn}; (f) $F1$ on the dataset D_{bconn}.

We also evaluate the effectiveness of various traffic analysis

TABLE I
WF ATTACK RESULTS ON THE IMPLEMENTED DEFENSE SCHEMES.

	D_{conn}					D_{mconn}					D_{bconn}				
	BWO	AppSniffer		NeuTic		BWO	AppSniffer		NeuTic		BWO	AppSniffer		NeuTic	
		TPR	FPR	TPR	FPR		TPR	FPR	TPR	FPR		TPR	FPR	TPR	FPR
Undefended	0%	91.22%	0.01%	85.32%	0.05%	0%	93.85%	0.00%	87.36%	0.05%	0%	98.95%	0.01%	87.54%	0.02%
FRONT [30]	64.21%	41.32%	0.26%	42.70%	0.29%	69.23%	52.84%	0.28%	55.10%	0.35%	66.97%	56.28%	0.18%	65.50%	0.14%
WTF-PAD [7]	48.64%	47.92%	0.21%	33.64%	0.30%	56.32%	49.84%	0.34%	65.36%	0.33%	45.22%	63.12%	0.12%	72.18%	0.15%
Prism [29]	30.63%	73.02%	0.16%	85.22%	0.07%	32.68%	74.32%	0.21%	92.06%	0.12%	51.33%	69.30%	0.12%	94.74%	0.02%
Minipatch [33]	**11.42%**	97.00%	0.01%	96.22%	0.01%	**13.92%**	97.16%	0.01%	94.32%	0.03%	**18.97%**	97.32%	0.01%	93.34%	0.04%
STAP ($V = 7$)	56.29%	**22.08%**	**0.37%**	**17.88%**	**0.41%**	54.37%	**29.16%**	**0.58%**	**35.00%**	**0.56%**	64.19%	**38.68%**	**0.29%**	**48.96%**	**0.25%**
STAP ($V = 4$)	37.20%	36.32%	0.31%	21.04%	0.39%	30.13%	46.88%	0.43%	45.52%	0.45%	40.18%	51.60%	0.20%	56.76%	0.18%
STAP ($V = 2$)	23.16%	60.84%	0.19%	38.08%	0.30%	18.67%	68.32%	0.25%	61.16%	0.33%	25.41%	75.56%	0.11%	74.56%	0.12%

tools as discriminators, as shown in Figure 4. The rows represent the discriminators used for generating perturbations, while the columns display Acc and $F1$ of the corresponding traffic analysis tools. The results reveal that different discriminators perform comparably in generating effective perturbations that mitigate WF attacks across datasets. However, AppSniffer consistently outperforms NeuTic, owing to its ensemble architecture, which combines multiple classifiers to enhance performance and generalization. This adaptability enables AppSniffer to provide robust privacy protection against a diverse range of traffic analysis tools. Consequently, AppSniffer was selected as the discriminator in STAP to facilitate more effective adversarial training.

C. Comparative Analysis and Effectiveness

In this subsection, we conduct a comparative analysis of STAP against other WF defense schemes in an open-world scenario. The open-world scenario provides a complex and realistic environment for evaluating WF defense schemes, encompassing client visits to both monitored and unmonitored websites, some of which may be unknown to the attacker. In this scenario, the attacker's goal is to accurately identify which specific monitored website the client is visiting. We use two principal metrics, the TPR and the FPR, to quantify the effectiveness of WF defenses. The TPR offers insight into the accuracy of the classifier in correctly identifying visits to monitored websites. Conversely, the FPR reveals the classifier's tendency to incorrectly label visits to unmonitored websites as monitored. We consider three configurations of STAP: STAP ($V = 2$), STAP ($V = 4$), and STAP ($V = 7$), where V denotes the number of perturbation layers generated through STAP's adversarial training rounds, corresponding to the V-layered perturbation models utilized by STAP. These configurations facilitate a direct comparison of STAP's performance under varying levels of bandwidth overhead with other defense schemes. The results are presented in Table I.

Table I shows that both AppSniffer and NeuTic demonstrate strong detection capabilities on undefended traffic among three datasets, achieving at least an 85% TPR in identifying visits to any monitored website. This serves as the baseline for evaluating the effectiveness of different WF defense schemes. Among comparative schemes, STAP ($V = 7$) achieves the best defense performance, followed by STAP ($V = 4$) as the second-best. It is known that FRONT incurs the highest bandwidth overhead across all three datasets. On average, STAP ($V = 4$) achieves a nearly 10% reduction in TPR compared to FRONT while reducing bandwidth overhead by 30% across all datasets and against two traffic classification tools. Notably, when evaluated with the dataset D_{conn} and the NeuTic traffic analysis tool, STAP ($V = 4$) achieves a more substantial 22% TPR decrease. Additionally, WTF-PAD leverages adaptive obfuscation strategies to reduce bandwidth overhead compared to FRONT. Compared to WTF-PAD, STAP ($V = 4$) optimizes performance by reducing bandwidth overhead by 15% while improving the TPR by an average of 10%.

Minipatch and Prism, however, exhibit limited defense effectiveness. To provide a comparison, we introduce STAP ($V = 2$). On average, STAP ($V = 2$) maintains 8% higher bandwidth than Minipatch and 15% lower bandwidth overhead than Prism, while achieving 30% and 20% lower TPR compared to Minipatch and Prism, respectively. It is noteworthy that when defending against NeuTic, Prism's poor performance is comparable to that of undefended traffic, whereas Minipatch inadvertently makes the traffic more discernible. These findings can be attributed to their perturbation strategies. Minipatch, aiming for minimal overhead, inserts identical dummy packets across similar websites. However, this predictability can unintentionally facilitate traffic identification, particularly under adversarial WF attacks. Prism attempts to obscure traffic by varying packet lengths, which slightly improves FPR but does not significantly affect the TPR. The minimal variance in payload lengths of resource packets results in a limited feature space, which undermines Prism's efficacy and utility in enhancing privacy.

Overall, STAP outperforms in defense efficacy, marked by the lower TPR and higher FPR. STAP's superiority lies in its multi-layered perturbation models, optimized for selecting different various morphing website targets, which disrupt traffic analysis with intricate transmission patterns. Additionally, STAP's iterative framework enhances adaptability, enabling adjustments in the number of perturbation models. This flexibility allows for tailoring the defense requirement to specific needs by varying the bandwidth overhead and defense strength, a feature not provided by other schemes.

Given the inherent imbalance between monitored and unmonitored websites in all three datasets, interpreting WF defense schemes based solely on TPR and FPR requires

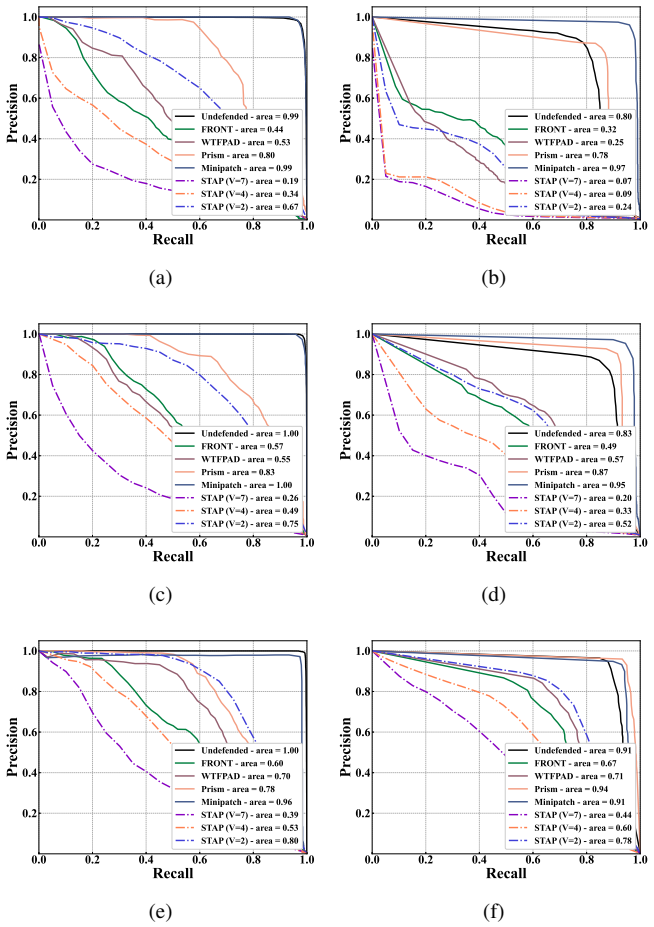

Fig. 5. Precision-Recall curves: (a) AppSniffer on the dataset D_{conn}; (b) NeuTic on the dataset D_{conn}; (c) AppSniffer on the dataset D_{mconn}; (d) NeuTic on the dataset D_{mconn}; (e) AppSniffer on the dataset D_{bconn}; (f) NeuTic on the dataset D_{bconn}.

caution. This imbalance can lead to inadequate explanations due to the base-rate fallacy, as mentioned in [7] and [38]. To address this concern and provide a more comprehensive analysis of WF defense schemes in the open-world scenario, we incorporate precision and recall curves (P-R curves) into our evaluation. In the P-R curves, a curve nearing the bottom left corner signifies better defense performance. Utilizing the area under curve (AUC) concept from ROC curves, the area beneath the P-R curve serves as a performance metric. The results are represented in Figure 5.

Across all three datasets, STAP consistently demonstrates its effectiveness as a robust WF defense scheme. STAP ($V = 4$) achieves an optimal balance between defense effectiveness and efficiency, outperforming four comparative schemes across both traffic analysis classifiers. In another, STAP ($V = 2$) offers a more bandwidth-efficient option with reasonable defense capabilities, STAP ($V = 7$) enhances the defense performance further at the cost of additional bandwidth overhead. Additionally, WTF-PAD and FRONT exhibit similar defensive performances, aligning with findings from previous experiments. Minipatch proves to be less effective, showing an increased susceptibility to WF attack identification, a vulnerability that is particularly pronounced when evaluated with NeuTic. Similarly, Prism exhibits a high identification rate, closely resembling undefended traffic scenarios when assessed against NeuTic, thereby emphasizing its limited defensive efficacy.

D. Information Leakage Measurement

In this subsection, we quantitatively assess the effectiveness of various WF defenses from the perspective of information leakage. We employ the WeFDE framework [39], an information theory-based approach, to estimate the mutual information between specific features and websites that could be exploited as website fingerprinting. Higher information leakage indicates a less effective defense against potential attacks, even if the defense achieves low attack accuracy on known traffic analysis tools.

Consistent with prior works [30], [40], we compute the information leakage for all 3043 features within the dataset D_{mconn}, excluding redundant ones, and select the top 100 non-redundant features that leak the most bits of information. We then calculate the mean and variance of their leakage. Additionally, we present the bandwidth overhead associated with each defense scheme. The results are illustrated in Figure 6.

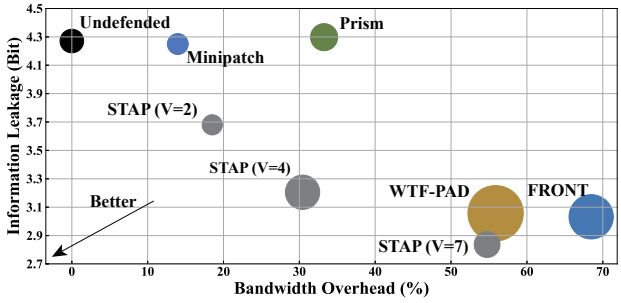

Fig. 6. The average and variance of information leakage over the top-100 informative features, along with the bandwidth overhead. The size of the marker represents the magnitude of the variance for each defense scheme.

From the figure, it is evident that both Minipatch and Prism provide defense against WF attacks without significantly reducing information leakage, indicating that these defenses are relatively transparent from an information leakage standpoint. Furthermore, WTF-PAD and FRONT exhibit similar defense performance to STAP ($V = 7$), but WTF-PAD shows higher information leakage and greater variance, while FRONT involves higher bandwidth consumption. STAP demonstrates the least information leakage while achieving better defense effectiveness than existing methods, with an acceptable bandwidth overhead.

E. Manipulation Efficiency

In this subsection, we analyze the number of manipulations performed on each downstream packet to discuss the potential latency introduced by various WF defense schemes. In real-time traffic scenarios, WF defenses modify each arriving packet to alter its spatio-temporal characteristics based on pre-generated packet manipulation strategies. For the purposes of our analysis, we define each manipulation as an individual

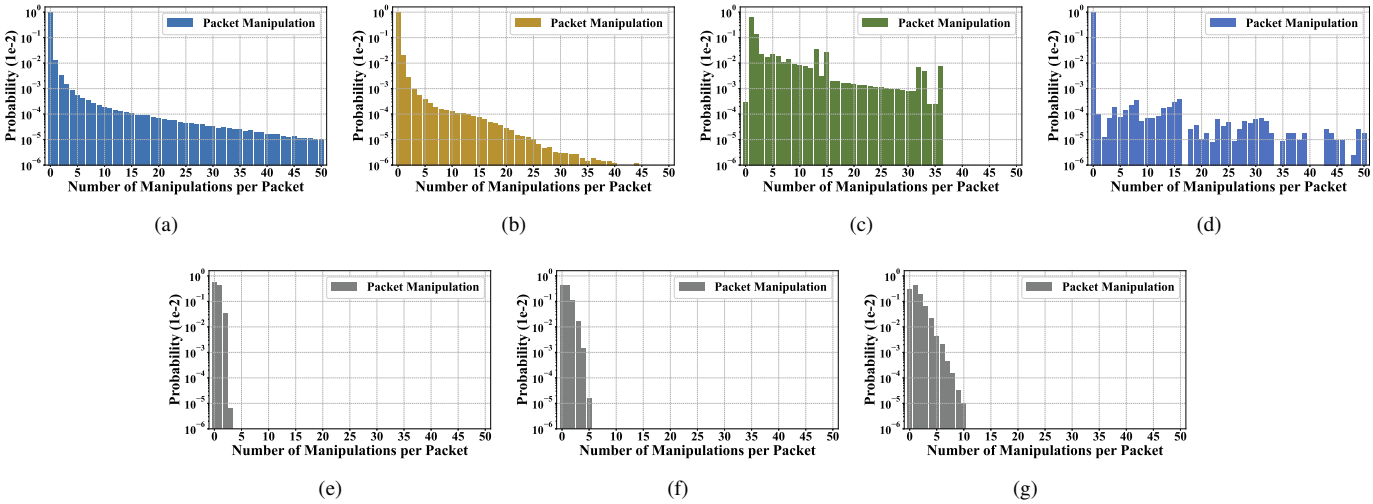

Fig. 7. Statistical distribution of the number of manipulations per downstream packet: (a) FRONT; (b) WTF-PAD; (c) Prism; (d) Minipatch; (e) STAP ($V = 2$); (f) STAP ($V = 4$); (g) STAP ($V = 7$).

packet operation, including packet padding, packet insertion, and packet splitting. The number of manipulations per packet significantly impacts latency, as more manipulations can cause increased congestion for subsequent packets. Using the dataset D_{mconn}, we count the number of manipulations for each downstream packet and present the statistical distribution in Figure 7. We only display data for packets undergoing up to 50 manipulations, as this range is sufficient for discussion.

Unlike other WF defense schemes, STAP requires fewer manipulations per packet and maintains a balance between manipulated and non-manipulated packets. Specifically, STAP ($V = 2$) has a manipulation count ranging from 0 to 3, STAP ($V = 4$) ranges from 0 to 5, and STAP ($V = 7$) ranges from 0 to 10. This efficiency is due to STAP's ability to select similar website targets and capture intrinsic similarities in the target's latent resource-states, thereby implementing effective obfuscation with fewer manipulations.

In contrast, other defense schemes use different strategies. FRONT inserts numerous packets into the initial traffic segment, resulting in a high proportion of non-manipulated packets. However, this also leads to higher latency, with a considerable number of packets undergoing more than 50 manipulations. Similarly, WTF-PAD and Minipatch, although not manipulating every packet, require an impractical number of additional packets per downstream packet. Prism leverages the frequency characteristics of spatio-temporal features by splitting MPS packets into multiple non-MPS packets, resulting in fewer unmanipulated packets, with the number of manipulated packets ranging from 1 to 36.

Additionally, we conduct practical deployment experiments using cloud server hosting HTTP/3 websites, integrating STAP and four compared WF defense schemes into *quic-go* [41] for real-time obfuscation during resource transmission. Over three days, we collect 60 response time samples for each scheme. We compute the additional time required to load website resources using the defense schemes compared to the undefended scenario. The results are presented in Figure 8.

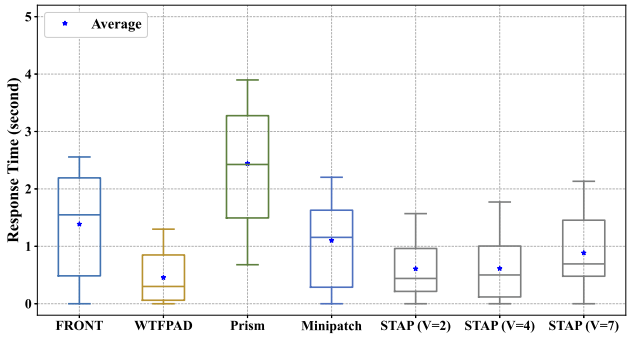

Fig. 8. Response time increase for WF defense schemes compared to the undefended scenario.

The results demonstrate that while STAP's response time increases with the number of perturbation layers, it remains within an acceptable range, with an average increase of less than 1 second, outperforming the other four WF defense schemes. Consistent with previous analyses, FRONT exhibits significantly higher response times due to its frequent packet manipulations, which lead to considerable congestion during transmission. WTF-PAD, although also employing frequent packet manipulations, achieves performance comparable to STAP by scheduling these operations during idle periods, thereby reducing their impact on response time. Minipatch shows inconsistent performance despite leveraging adversarial training to determine optimal manipulation positions. This inconsistency arises from its reliance on multiple manipulations at fixed positions, which can exacerbate congestion under certain network conditions. Prism incurs the highest response time among all the schemes. Its packet-by-packet manipulation strategy introduces substantial delays, as it requires waiting for packet arrivals and performing real-time calculations before transmission, which naturally increases latency in the communication process.

Overall, STAP minimizes operational overhead by requiring only a small number of manipulations per packet, thereby

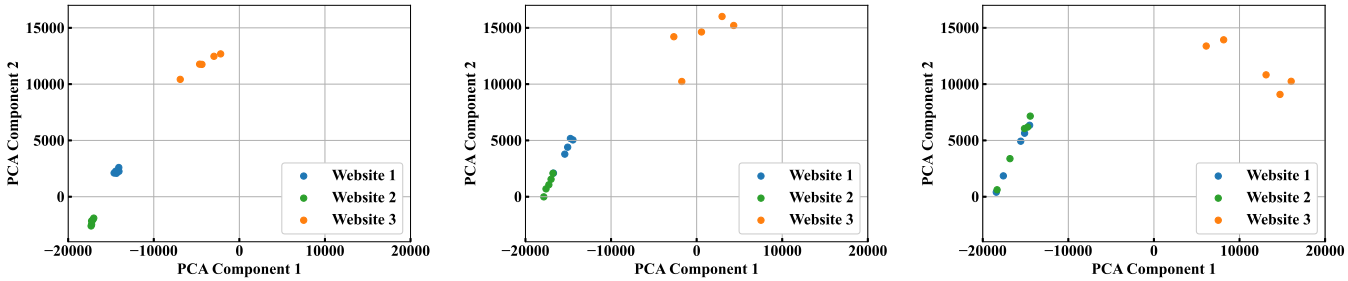

Fig. 9. Visual analysis through PCA: (a) Before perturbation; (b) During perturbation; (c) After perturbation.

maintaining efficiency and reducing latency.

F. Perturbation Variation

This subsection explores the impact of perturbation variation by applying perturbations across monitored websites to dissect the perturbation mechanism of STAP. For visual analysis, principal component analysis (PCA) is utilized to project all samples into a 2D feature space, and we also randomly select three websites for illustration, as shown in Figure 9.

Before perturbation, subplot (a) illustrates that samples within the same website are tightly clustered, distinctively separated from others, simplifying the task of website identification. The introduction of perturbations, following STAP's multi-layered perturbation framework, significantly alters the distribution within this feature space. Subplot (b) displays this shift, showcasing how perturbations incrementally broaden the feature space for each website, thereby obscuring the clear separations that once existed. The completion of the perturbation process, as shown in subplot (c), elucidates the strategy behind STAP. The overlap between the distributions of website 1 and website 2, combined with the marked dispersion in the distribution of website 3, highlights the effectiveness of the perturbation. Such merging and expansion of website distributions effectively reduce website fingerprinting, validating STAP's effectiveness in defending against adversarial WF attacks.

G. Future Work

The experimental results demonstrate the effectiveness of STAP in achieving robust defense performance compared to four advanced WF defense schemes. To further advance WF defenses, we propose two promising research directions worthy of exploration and discussion:

1) Enhancing Packet Manipulations: While STAP leverages two packet operations to modify traffic patterns, incorporating additional manipulations, such as packet splitting and merging, could achieve deeper traffic obfuscation by further disrupting the spatio-temporal characteristics of traffic. However, implementing these advanced manipulations requires the development of more sophisticated perturbation models capable of effectively coordinating operations while maintaining a balance between bandwidth overhead and defense performance.

2) Leveraging Multiple Discriminators: STAP's adversarial training process relies on a single discriminator to guide the perturbation models. To provide general privacy protection against a broader range of traffic classifiers and mitigate the potential impact of inaccurate feedback from a single discriminator, introducing multiple discriminators, each specializing in different traffic patterns, could enhance obfuscation effectiveness and generalization. This approach would diversify feedback during training and improve STAP's adaptability to evolving WF attacks.

VI. Conclusion

In this study, we introduce STAP, a novel asymmetric defense scheme that alters packets in traffic flows using multi-layered perturbation models. It can operate independently on the server-side without requiring user cooperation, ensuring end-to-end obfuscation between users and servers. The inherent encryption capabilities and frame design of QUIC facilitate such manipulations naturally and unnoticeably. STAP demonstrates effectiveness against state-of-the-art traffic analysis tools, offering better defense performance with less bandwidth overhead compared to other advanced WF defense schemes. With the increasing adoption of HTTP/3 in web services and the growing demand for privacy protection in resource-constrained client devices, STAP shows great potential for application in these scenarios.

References

[1] S. N. Han and N. Crespi, "Semantic service provisioning for smart objects: Integrating iot applications into the web," *Future Gener. Comput. Syst.*, vol. 76, pp. 180–197, 2017.

[2] M. Shen, K. Ji, J. Wu, Q. Li, X. Kong, K. Xu, and L. Zhu, "Real-time website fingerprinting defense via traffic cluster anonymization," in *Proc. IEEE Symp. Secur. Privacy*, 2024, pp. 3238–3256.

[3] K. M. A. Kamal and S. Almuhammadi, "Vulnerability of virtual private networks to web fingerprinting attack," in *Advances in Security, Networks, and Internet of Things*, 2021, pp. 147–165.

[4] M. Shen, K. Ye, X. Liu, L. Zhu, J. Kang, S. Yu, Q. Li, and K. Xu, "Machine learning-powered encrypted network traffic analysis: a comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 25, pp. 791–824, 2022.

[5] P. Liu, L. He, and Z. Li, "A survey on deep learning for website fingerprinting attacks and defenses," *IEEE Access*, vol. 11, pp. 26 033–26 047, 2023.

[6] 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. Security*, 2014, pp. 227–238.

[7] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an efficient website fingerprinting defense," in *Proc. Eur. Symp. Res. Comput. Security*, 2016, pp. 27–46.
[8] M. S. Rahman, M. Imani, N. Mathews, and M. Wright, "Mockingbird: Defending against deep-learning-based website fingerprinting attacks with adversarial traces," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 1594–1609, 2021.
[9] "Daita: Defense against ai-guided traffic analysis," accessed: March 18, 2025. [Online]. Available: https://mullvad.net/en/vpn/daita.
[10] T. Wang and I. Goldberg, "Walkie-talkie: An efficient defense against passive website fingerprinting attacks," in *Proc. USENIX Security Symp.*, 2017, pp. 1375–1390.
[11] G. Cherubin, J. Hayes, and M. Juárez, "Website fingerprinting defenses at the application layer," in *Proc. Privacy Enhancing Technol.*, 2017, pp. 168–185.
[12] J. Gong, W. Zhang, C. Zhang, and T. Wang, "Wfdefproxy: Real world implementation and evaluation of website fingerprinting defenses," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 1357–1371, 2024.
[13] M. Nasr, A. Bahramali, and A. Houmansadr, "Deepcorr: Strong flow correlation attacks on tor using deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2018, pp. 1962–1976.
[14] S. Chen, R. Wang, X. Wang, and K. Zhang, "Side-channel leaks in web applications: A reality today, a challenge tomorrow," in *Proc. IEEE Symp. Secur. Privacy*, 2010, pp. 191–206.
[15] L. Qiao, B. Wu, S. Yin, H. Li, W. Yuan, and X. Luo, "Resisting dnn-based website fingerprinting attacks enhanced by adversarial training," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 5375–5386, 2023.
[16] C. V. Kopek, E. W. Fulp, and P. S. Wheeler, "Distributed data parallel techniques for content-matching intrusion detection systems," in *Proc. IEEE Military Comm. Conf.*, 2007, pp. 1–7.
[17] M. Husák, M. Čermák, T. Jirsík, and P. Čeleda, "Https traffic analysis and client identification using passive ssl/tls fingerprinting," *EURASIP J. Inf. Security*, vol. 2016, pp. 1–14, 2016.
[18] C. Liu, Z. Cao, G. Xiong, G. Gou, S.-M. Yiu, and L. He, "Mampf: Encrypted traffic classification based on multi-attribute markov probability fingerprints," in *Proc. IEEE/ACM 26th Int. Symp. Qual. Service (IWQoS)*, 2018, pp. 1–10.
[19] B. Anderson, S. Paul, and D. McGrew, "Deciphering malware's use of tls (without decryption)," *J. Comput. Virol. Hacking Techn.*, vol. 14, pp. 195–211, 2018.
[20] T. Van Ede, R. Bortolameotti, A. Continella, J. Ren, D. J. Dubois, M. Lindorfer, D. Choffnes, M. Van Steen, and A. Peter, "Flowprint: Semi-supervised mobile-app fingerprinting on encrypted network traffic," in *Proc. Netw. Distrib. Syst. Security Symp.*, 2020, pp. 1–18.
[21] C. Dong, Z. Lu, Z. Cui, B. Liu, and K. Chen, "Mbtree: Detecting encryption rats communication using malicious behavior tree," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 3589–3603, 2021.
[22] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "Fs-net: A flow sequence network for encrypted traffic classification," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2019, pp. 1171–1179.
[23] X. Yun, Y. Wang, Y. Zhang, C. Zhao, and Z. Zhao, "Encrypted tls traffic classification on cloud platforms," *IEEE/ACM Trans. Net.*, vol. 31, pp. 164–177, 2022.
[24] X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi, and J. Yu, "Et-bert: A contextualized datagram representation with pre-training transformers for encrypted traffic classification," in *Proc. ACM Web Conf. (WWW)*, 2022, pp. 633–642.
[25] S. Oh, M. Lee, H. Lee, E. Bertino, and H. Kim, "Appsniffer: Towards robust mobile app fingerprinting against vpn," in *Proc. ACM Web Conf. (WWW)*, 2023, pp. 2318–2328.
[26] 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*, 2012, pp. 332–346.
[27] X. Cai, R. Nithyanand, and R. Johnson, "Cs-buflo: A congestion sensitive website fingerprinting defense," in *Proc. 13th Workshop Privacy Electron. Soc.*, 2014, pp. 121–130.
[28] J. K. Holland and N. Hopper, "Regulator: A straightforward website fingerprinting defense," in *Proc. Privacy Enhancing Technol.*, 2022, pp. 344–362.
[29] W. Li, X.-Y. Zhang, H. Bao, B. Yang, Z. Li, H. Shi, and Q. Wang, "Prism: Real-time privacy protection against temporal network traffic analyzers," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 2524–2537, 2023.
[30] J. Gong and T. Wang, "Zero-delay lightweight defenses against website fingerprinting," in *Proc. USENIX Security Symp.*, 2020, pp. 717–734.
[31] M. Nasr, A. Bahramali, and A. Houmansadr, "Defeating dnn-based traffic analysis systems in real-time with blind adversarial perturbations," in *Proc. USENIX Security Symp.*, 2021, pp. 2705–2722.
[32] S. Shan, A. N. Bhagoji, H. Zheng, and B. Y. Zhao, "Patch-based defenses against web fingerprinting attacks," in *Proc. 14th ACM Workshop Artif. Intell. Secur.*, 2021, pp. 97–109.
[33] D. Li, Y. Zhu, M. Chen, and J. Wang, "Minipatch: Undermining dnn-based website fingerprinting with adversarial patches," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 2437–2451, 2022.
[34] 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. Security*, 2020, pp. 1971–1985.
[35] M. Trevisan, F. Soro, M. Mellia, I. Drago, and R. Morla, "Attacking doh and ech: Does server name encryption protect users' privacy?" *ACM Trans. Internet Technol.*, vol. 23, pp. 1–22, 2023.
[36] P. S. Efraimidis and P. G. Spirakis, "Weighted random sampling with a reservoir," *Informat. Process. Lett.*, vol. 97, pp. 181–185, 2006.
[37] J.-P. Smith, L. Dolfi, P. Mittal, and A. Perrig, "Qcsd: A quic client-side website-fingerprinting defence framework," in *Proc. USENIX Security Symp.*, 2022, pp. 771–789.
[38] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at internet scale," in *Proc. Netw. Distrib. Syst. Security Symp.*, 2016, pp. 1–15.
[39] S. Li, H. Guo, and N. Hopper, "Measuring information leakage in website fingerprinting attacks and defenses," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2018, pp. 1977–1992.
[40] J. Gong, W. Zhang, C. Zhang, and T. Wang, "Surakav: Generating realistic traces for a strong website fingerprinting defense," in *Proc. IEEE Symp. Secur. Privacy*, 2022, pp. 1558–1573.
[41] "A quic implementation in pure go," accessed: March 18, 2025. [Online]. Available: https://github.com/quic-go/quic-go.

Jia-Nan Huang received the B.E. degree from Electronic and Information Engineering in Zhejiang Agriculture and Forestry University in 2021. He is currently working towards the Ph.D. degree in Nanjing University of Science and Technology. His research interests include network traffic obfuscation, network traffic analysis, and network covert channels.

Weiwei Liu (Member, IEEE) received the B.S. degree in automation and the Ph.D. degree in control science and engineering from the Nanjing University of Science and Technology in 2010 and 2015, respectively. From 2014 to 2015, he was a Visiting Scholar with the Department of Computer Science, University of California at Davis. He is currently an Associate Professor with Nanjing University of Science and Technology. His research interests include encrypted traffic analysis and traffic obfuscation.

Guangjie Liu (Member, IEEE) received the B.S. degree in electrical and computer engineering and the Ph.D. degree in control science and engineering from Nanjing University of Science and Technology in 2002 and 2007, respectively. From 2016 to 2017, he was a Visiting Scholar with the Department of Computer Science, University of California at Davis. He is currently a Professor with Nanjing University of Information Science and Technology. His research interests are network and information security.

Bo Gao received the B.S. degree in physics from Jilin University, Jilin, in 2016, and the M.Eng. degree in control engineering from Nanjing University of Science and Technology, Nanjing, China, in 2020. He is currently pursuing the Ph.D. degree in Nanjing University of Science and Technology. His research interests include anonymity networks and network traffic analysis.

Fengyuan Nie received the B.S. degree in automation from Shanghai University of Electric Power in July 2020. He is currently working towards the Ph.D. degree in Nanjing University of Science and Technology. His research interests include internet of things, network traffic classification and deep learning.

Marco Mellia (Fellow, IEEE) is a full professor at PoliTO, Italy. He coordinates the SmartData@PoliTO centre, an interdisciplinary lab focusing on machine learning, data science and applications to network management and cybersecurity. He has co-authored over 250 papers published in international journals and leading conferences. He won the IRTF ANR Prize at IETF-88, and many best paper awards. He is the Editor-in-Chief of the PROCEEDINGS OF THE ACM ON NETWORKING.