Analyzing the Impact of Encryption on Traffic Classification through Explainable AI

Davide Di Monda*†, Alfredo Nascita*, Raffaele Carillo*, Antonio Pescapé*
*University of Napoli Federico II, Napoli (Italy), †IMT School for Advanced Studies, Lucca (Italy)
{davide.dimonda, alfredo.nascita, raffaele.carillo, pescape}@unina.it

***Abstract*—Traffic Classification (TC) plays a crucial role in modern network management and monitoring. The increasing adoption of encrypted protocols—most notably TLS 1.3—poses serious challenges to traditional TC methods as it reduces the visibility of traditionally employed traffic features. At the same time, Few-Shot Learning (FSL) models are emerging as a promising solution for low-data TC scenarios (as in the case of new encrypted protocols), yet their lack of interpretability remains an open issue, especially under encrypted conditions. In this work, we study the impact of encryption on both the performance and interpretability of FSL-based traffic classifiers. Relying on a recently proposed FSL model, META MIMETIC, we simulate the progressive removal of plaintext features and assess the impact on TC. We then apply Integrated Gradients, a well-known eXplainable AI technique, to analyze how encryption alters byte-level feature importance. Our findings shed light on how encryption reshapes model behaviors and the relevance of input features for classification.**

***Index Terms*—traffic classification, eXplainable AI, encrypted traffic, few-shot learning**

I. Introduction

Traffic Classification (TC) is essential for modern network management, enabling efficient, secure, and high-performance data handling by supporting key tasks such as quality of service enforcement and dynamic load balancing. Mobile-app TC, in particular, has become increasingly important with the global rise in smartphone use [1]. However, it remains challenging due to the diversity of apps, frequent updates, and widespread encryption. By 2025, HTTPS is expected to account for 99% of traffic [2], making SSL/TLS flows difficult to inspect [3]. These challenges have driven the adoption of *Machine Learning* (ML) and, more recently, *Deep Learning* (DL) classifiers [4]. Yet, DL still face two key issues: (*i*) dependence on large, up-to-date labeled datasets, which are often costly to obtain; and (*ii*) degraded performance under *traffic shift*, as traffic patterns evolve, potentially due to the increasing adoption of new encryption protocols. To tackle these limitations, *Few-Shot Learning* (FSL) paradigm has emerged, enabling rapid generalization to new TC tasks using limited labeled data and prior knowledge.

This work is partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART") and the "xInternet" Project within the PRIN 2022 program (D.D.104-02/02/2022), funded by MUR. This manuscript reflects only the authors' views and opinions and the Ministry cannot be considered responsible for them.

FSL enables training on prior knowledge from abundant (historical) data and adapting to new tasks with only a few samples, typically tens of bidirectional flows [5]. Nonetheless, the body of literature investigating FSL in the domain of network TC is still in its infancy [6]. In addition, despite the promise of FSL techniques in low-data settings, a major challenge remains largely unexplored: the lack of interpretability in the decisions made by FSL-based traffic classifiers. These models often function as black boxes, making it difficult to understand which features or parts of the input drive their predictions, raising concerns about trustworthiness and robustness and highlighting the urgent need for explainability analyses [7]. In this context, eXplainable Artificial Intelligence (XAI) aims to elucidate the decision-making processes of AI-based tools. This problem is further exacerbated by the growing adoption of modern encryption protocols, particularly TLS 1.3, which encrypts large portions of the handshake packets—including the Server Name Indication (SNI) extension. Understanding how these models operate under such conditions is critical. For this reason, we evaluate the effects of encryption to understand its impact on classification performance, and we conduct explainability analysis to investigate how it alters the importance of payload bytes used as input. Our analysis is conducted on META MIMETIC [6], a novel FSL solution that leverages the meta-learning paradigm to optimize multimodal embedding functions, enabling effective learning from both structural and content-based input representations. In this study, ***(i)*** we quantify TLS 1.3 support and adoption in real-world datasets. Then, ***(ii)*** we simulate the effects of increasing encryption to understand its impact on TC performance in few-shot regime. Finally, ***(iii)*** we investigate the impact of encryption adoption through a XAI methodology based on the well-known Integrated Gradients technique.

The rest of this paper is organized as follows. Sec. II reviews related studies. Sec. III outlines the design of META MIMETIC and the used XAI techniques. Sec. IV details the experimental setup. Sec. V examines performance under varying encryption levels and presents XAI-based insights. Finally, Sec. VI concludes and discusses future directions.

II. Related Works

Encrypted TC has gained attention in recent years especially with the rise of DL techniques. To address the data-hungry nature of these models, recent works have explored FSL. This is especially relevant for the constantly evolving mobile-app

ISBN 978-3-903176-72-0 © 2025 IFIP

traffic, which is difficult to label due to privacy and operational constraints.

Within the current state of the art, the explainability of FSL for TC have only been marginally addressed. Zhang et al. [8] propose a method for training a generative diffusion model using a limited number of traffic samples to produce synthetic data. This data is then used to train an intrusion detection system that integrates convolutional networks with bidirectional gated recurrent units. The interpretability of the model is analyzed using SHAP to identify the most influential traffic features. Cerasuolo et al. [9] combine FSL and Class Incremental Learning, introducing SWEET, a novel explainable approach for mobile TC, which uses augmentation strategies to address the challenges of app evolution and data scarcity. Additionally, they combine visualization (t-SNE), similarity (nearest-neighbor similarity), and attribution-based (DeepSHAP) explanations to gather deeper insight on how the decisions of a classifier vary across different apps scenarios. While FSL enables rapid adaptation to new traffic classes, it is also important to explicitly consider *traffic shift* in the existing classes (i.e. those caused by the adoption of new encrypted protocols). To the best of our knowledge, only Yang et al. [10] address this issue, evaluting their FSL approach on a dataset containing traffic from apps with a newer version from the one used for training. However, their work does not include any interpretability analysis of traffic shifts and does not explicitly consider shifts caused by encryption.

Positioning. This work builds on the findings of [6], which proposed META MIMETIC—a novel solution designed to improve the classification of encrypted mobile-app traffic in few-shot scenarios by using a multimodal embedding function that extracts features from multiple views of the same traffic object. In contrast to previous literature, we focus on traffic shifts caused by the adoption of more secure protocols (i.e. TLS 1.3) and analyze how such changes affect META MIMETIC performance. In particular, we aim at interpreting META MIMETIC's behavior under different encryption conditions through XAI techniques to shed light on its black-box nature.

III. METHODOLOGY

A. Background on Meta Learning

Meta-learning aims to improve the classification accuracy of a model on a task $\mathcal{T}$ through the meta-knowledge extracted across a distribution of tasks $p(\mathcal{T})$, implementing a *learning-to-learn* paradigm [11].

Meta learning is typically structured into three phases: (*i*) *Meta-training* builds the prior knowledge of a model by leveraging classes with abundant data; (*ii*) *Meta-validation* is used to monitor the model performance at the end of each meta-training epoch; (*iii*) *Meta-testing* focuses on classes with limited available data, simulating the few-shot operational condition. In a FSL scenario, each phase uses distinct datasets with *non-overlapping label spaces*, denoted as $\mathcal{D} = \{\mathcal{D}_{nf}, \mathcal{D}_{nf2}, \mathcal{D}_f\}$. Going into detail, meta-training simulates real-world conditions—where only few samples are available—by sampling tasks $\mathcal{T}_{nf} \sim p(\mathcal{T})$, each with limited per-class data. During each epoch, multiple tasks are drawn, each defined by a training (viz. *support*) set and a test (viz. *query*) set, both sampled from $\mathcal{D}_{nf}$. Each task involves N classes (*ways*), with $N \times K_s$ support samples (where $K_s = K$ is called *shots*) and $N \times K_q$ query samples (i.e. such task is called N-way K-shot). Meta-testing and meta-validation evaluate generalization on new tasks $\mathcal{T}_f$ and $\mathcal{T}_{nf2}$, which are drawn from $\mathcal{D}_f$ and $\mathcal{D}_{nf2}$, respectively. As in meta-training, each task uses the support set to classify samples in the query set. The model performance is measured by averaging the achieved per-task scores on the query sets.

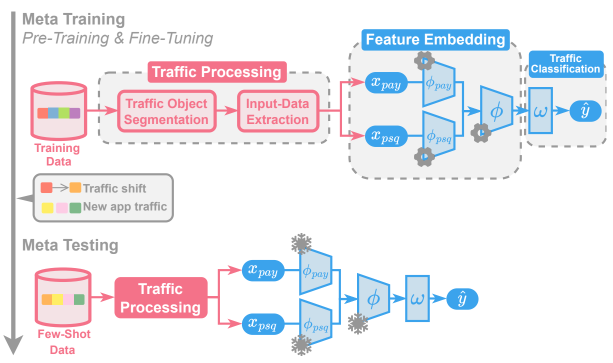


Fig. 1: Workflow of META MIMETIC for encrypted mobile app TC. After meta-training on abundant data, traffic is pre-processed and used to train modality-specific embeddings. During meta-testing, the model adapts to traffic shift with frozen embeddings.

B. Meta Mimetic Traffic Classification Workflow

Fig. 1 shows the general workflow and components of META MIMETIC, which are presented in detail below.

Traffic Processing. In line with recent literature [12, 13], network traffic is initially segmented into traffic objects (i.e. *traffic object segmentation* phase), and, specifically, packets are grouped into bidirectional flows—i.e. *biflows*.[1] During the *input-data extraction* phase, we derive two input types from each biflow: *packet-field sequences* (PSQ) and *payload bytes* (PAY). The input PSQ is made up of a set of 4 informative unbiased features extracted from the sequence of the first N_p packets of a biflow. In particular, the selected features are: (*i*) PL, the number of bytes in the transport layer payload; (*ii*) DIR, the packet direction (can be -1 or 1); (*iii*) WIN, the TCP window size (equal to 0 for UDP packets); (*iv*) IAT, the elapsed time since the arrival of the previous packet (i.e. the inter-arrival time). For PAY input, we extract the first N_b bytes of the L4-layer payload data for each biflow. We apply appropriate zero-padding when a biflow contains fewer than N_p packets or N_b bytes. Lastly, both PSQ and PAY are transformed using a Min-Max normalization. It is worth noting that the effectiveness of PAY is threatened by ubiquitous traffic encryption [14], in particular by TLS 1.3. In this new version

[1]A biflow is defined as a stream of packets sharing the same 5-tuple (i.e. transport-level protocol, source and destination IP addresses and ports) regardless of the direction of communication.

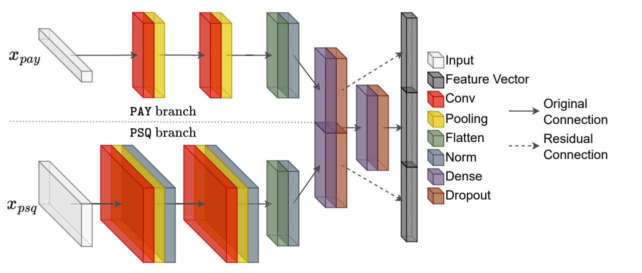


Fig. 2: Architecture of META MIMETIC embedding function with Residual Connections. Each branch is fed with the related input type (i.e. PAY or PSQ).

of TLS, several fields that were previously in plaintext—e.g., the SNI extension—are now encrypted.

Feature Embedding. The inputs are then processed by the multimodal embedding function, as illustrated in Fig. 2. META MIMETIC comprises two branches, each processing a different modality of the input. The first branch (ϕ_{pay}), fed with the PAY input, consists of two 1D convolutional layers, each followed by a 1D max-pooling layer, and concludes with a dense layer. The second branch (ϕ_{psq}), fed with the PSQ input, includes two 2D convolutional layers interleaved with 2D max-pooling and batch normalization. The features extracted from both modalities are concatenated and processed further through a *shared dense layer* (ϕ). The specific layer hyperparameters are those presented in [13] for the PAY branch and shared layer and in [15] for the PSQ branch. Next, the output of the shared layer is concatenated with the outputs from each branch (i.e. *Residual Connections* [6]), resulting in a feature vector called $\boldsymbol{\nu}$.

Traffic Classification. The traffic classifier $\omega(\boldsymbol{\nu})$ accepts the feature vector $\boldsymbol{\nu}$ as input and outputs a probability vector $\hat{\boldsymbol{y}}$ over the $\mathcal{C}$ classes. The classifier used in META MIMETIC is based on *Matching Networks* [11] (MatchingNet), which generalizes the nearest-neighbor classifier. Specifically, after embedding both the support set and the query biflow, the classifier $\omega(\cdot)$ computes $\hat{\boldsymbol{y}}$ using the formula: $\hat{\boldsymbol{y}} = \sum_{i=1}^{K_s} a(\boldsymbol{\nu}(\boldsymbol{x}_{query}), \boldsymbol{\nu}(\boldsymbol{x}_i))\boldsymbol{y}_i$, where K_s is the number of samples in the support set; $a(\cdot,\cdot)$ is an attention mechanism that measures the similarity between the embedding of the query biflow $\boldsymbol{\nu}(\boldsymbol{x}_{query})$ and each embedded support biflow $\boldsymbol{\nu}(\boldsymbol{x}_i)$; $\boldsymbol{y}_i$ is the class label of the i^{th} support sample. Finally, $\boldsymbol{x}_{query}$ is assigned the class of the support sample that is most similar to it according to the attention mechanism.

Learning Procedure. META MIMETIC is trained through a two-step training procedure. During *pre-training*, we train the two branches, ϕ_{pay} and ϕ_{psq}, separately to *refine their representation of the two input views*. In detail, we start by updating the current branch ϕ_m, where $m = \{pay, psq\}$, using a large number of N-way K-shot tasks drawn from $\mathcal{D}_{nf}$. The goal is to obtain a ϕ_m capable of structuring the embedding space around class similarity. The classifier labels the query set biflows through the support set and outputs softmax probability vectors $\hat{\boldsymbol{y}}$, which are compared with ground truth labels $\boldsymbol{y}$ to minimize categorical cross-entropy loss. After updating the weights of ϕ_m via backpropagation, meta-validation evaluates the early-stopping condition. After reaching the maximum number of epochs or satisfying the early-stopping condition, we froze all the layers of ϕ_{pay} and ϕ_{psq} except their final dense layers. Subsequently, during *fine-tuning*, we train only the unfrozen layers of ϕ_{pay} and ϕ_{psq} and the shared layer, ϕ, to refine the processing of the views of concatenated traffic objects. In such a phase, meta-training and -validation proceed as in the pre-training phase, except that both inputs, PAY and PSQ, are used simultaneously. Once fine-tuning is ended, the model can be adapted to few-shot classes (i.e. those in $\mathcal{D}_f$) via meta-testing. In this phase, the embedding function is frozen, and classification relies on comparing new biflows to the support set. Since the model was trained around class similarity, it can accurately recognize new traffic types given only a few biflows.

C. Interpreting FSL-based Traffic Classifiers with Integrated Gradients

Integrated Gradients [16] (IG) is an interpretability technique that relies on the concept of an input *baseline*. It is based on counterfactual intuition and compares an effect, e.g., a classifier's decision, to its absence, modeled by a baseline input. The IG value ϕ_m represents the importance of the m^{th} input in shifting the confidence for a given class relative to this baseline[2] (along the i^{th} input dimension), providing details on its contribution (either positive or negative) to model predictions. Detailing, IG is obtained by considering the straight-line path from the baseline ($\boldsymbol{x}'$) to the input ($\boldsymbol{x}$) and cumulating the gradients at all points along the path. Hence, given an input $\boldsymbol{x} = \{x_1, \dots, x_M\} \in \mathbb{R}^M$, the "effect" ϕ_m of each input x_m is given by:

$$\phi_m(\boldsymbol{x}) = \left(x_m - x_m^{'}\right) \times \int_{\alpha=0}^{1} \frac{\partial p_i\left(\boldsymbol{x}^{'} + \alpha\left(\boldsymbol{x} - \boldsymbol{x}^{'}\right)\right)}{\partial x_m} d\alpha \quad (1)$$

According to the *completeness* axiom, all the effects add up to the difference between the output of the model relative to the i^{th} class, at the input $\boldsymbol{x}$ and the baseline $\boldsymbol{x}'$:

$$\sum_{m=1}^{M} \phi_m(\boldsymbol{x}) = p_i(\boldsymbol{x}) - p_i(\boldsymbol{x}') \quad (2)$$

To explain the impact of encryption on our FSL-based traffic classifiers, the values to be explained are chosen as the soft-output associated with the i^{th} class, i.e. $p_i(\boldsymbol{x})$. As the focus is on analyzing the impact of encryption on the PAY input, we set the PSQ input to the baseline value prior to computing attributions. This ensures that its contribution is zero, allowing us to isolate and examine the role of the PAY input. In our methodology, we introduce *normalization* and *pooling* operations, as detailed in the following. In particular, when considering per-biflow (viz. local) explanations, we normalize the importance values by their sum. In this manner, we obtain importance measures independent of the specific soft-outputs

[2]We set the baseline for IG is set to all zero values for both inputs.

and a relative importance measure. In addition, our methodology involves the aggregation of local IG explanations, to provide more comprehensive global explanations.

IV. Experimental Setup

This section outlines the experimental setup, detailing the datasets and the hyperparameter settings of Meta Mimetic.

Datasets. We rely on two public datasets collected at the University of Napoli Federico II: `Mirage-2019` [17] and `Mirage-2022` [18]. `Mirage-2019`, collected between 2017–2019, includes $97k$ biflows from 40 Android apps. `Mirage-2022`, collected in 2021, contains $29k$ biflows from 9 apps, with class sizes ranging from 1866 to 5053 biflows. In more detail, we use `Mirage-2019` to train Meta Mimetic and a small number of labeled biflows (*25* per app) from `Mirage-2022` for meta-testing.

Hyperparameters Setup. The hyperparameters setup has been chosen according to preliminary experiments and results from previous studies [6]. Following the partition proposed in [19], we define three distinct sets of classes from `Mirage-2019` and `Mirage-2022`: (*i*) $\mathcal{C}_{nf}$ includes the 31 most populated apps from `Mirage-2019` (these apps are included in $\mathcal{D}_{nf}$ only); (*ii*) $\mathcal{C}_{nf2}$ comprises the remaining 9 apps from `Mirage-2019` (included in $\mathcal{D}_{nf2}$ only); (*iii*) $\mathcal{C}_{f}$ encompasses all 9 apps from `Mirage-2022` (included in $\mathcal{D}_{f}$ only). From $\mathcal{D}_{nf}$, $\mathcal{D}_{nf2}$, and $\mathcal{D}_{f}$, we sample the N-way K-shot tasks, where $N = |\mathcal{C}_f| = 9$, namely, equal to the number of apps in `Mirage-2022`, and $K = K_s = 25$. The number of query samples per app is set to $K_q = 100$. During each epoch of meta-training, we randomly sample 100 tasks. Similarly, 100 tasks are used for evaluating Meta Mimetic during meta-testing and meta-validation. For `MatchingNet`, we employ Euclidean distance as attention mechanism. Moreover, pre-training and fine-tuning phases of Meta Mimetic are conducted for 280 and 440 epochs, respectively. For traffic segmentation, we set $N_p = 10$ for `PSQ` and $N_b = 512$ for `PAY`, following the results from prior analyses [20].

Performance Metric. To evaluate the experimental results, we use the *macro F1-score*, which is computed as the average of the harmonic mean of Precision and Recall across all classes. The F1-score is reported as the per-task mean across the 100 sampled tasks.

V. Experimental Results

This section is organized as follows. We first analyze TLS 1.3 adoption in real-world datasets (Sec. V-A). Then, we simulate SNI encryption introducing `PAY` input variants and assess impact on classification (Sec. V-B). Finally, we use IG to examine how encryption affects byte-level feature importance (Sec. V-C).

A. TLS 1.3 Adoption Trends

Among the two input types used by Meta Mimetic, `PAY` —which contains raw payload data—is the most impacted by encryption and contains essential information for biflow classification in encrypted TCP connections using TLS. In TLS biflows, the first packet is named *Client Hello* (CH) and includes different extensions. Among these, the SNI extension, exposes the *server hostname* in plaintext, which can often be associated with the app generating the biflow and is commonly used as a proxy for TC tasks. The *TLS 1.3*, introduced in 2018, brings significant changes, including the encryption of the entire CH, rendering the SNI extension and its server hostname field ineffective for TC purposes. By analyzing the TLS biflows contained in the `Mirage-2019` and `Mirage-2022` datasets, we observe a substantial increase in support for TLS 1.3. Specifically, in `Mirage-2019` the percentage of TLS biflows with CH messages supporting TLS 1.3 is only 0.56%. In contrast, the percentage for the more recent `Mirage-2022` increases significantly, reaching 65.15%.

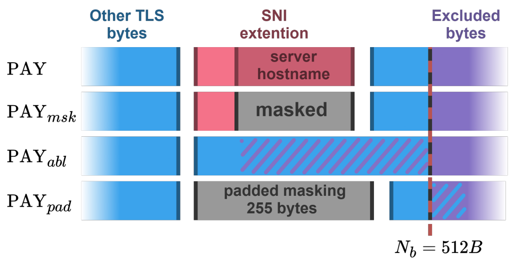


Fig. 3: Representation of *information-removal schemes*. SNI-extension bytes are colored in **red**, the masked ones are colored in **gray**, the other TLS bytes considered in the `PAY`$_*$ input are colored in **blue**, and those excluded from the `PAY`$_*$ input are in **violet**. Extra bytes considered in the `PAY`$_{abl}$ input (w.r.t. `PAY`) are shown with **violet hatched lines** and bytes ablated in the `PAY`$_{pad}$ input (w.r.t. `PAY`) are shown with **blue hatched lines**.

B. Simulating SNI Encryption: `PAY` Variants and Their Impact on Classification

Although we observed that TLS 1.3 is supported in a significant portion of `Mirage-2022` TLS biflows, the dataset still includes a substantial amount of TLS 1.2 traffic. Therefore, to simulate the gradual removal of SNI byte information, we investigate three `PAY` input variants. This results into *information removal schemes* [6] outlined in Fig. 3 and detailed below: (*i*) `PAY`$_{msk}$, where only the server hostname is *masked* and replaced by random values, preserving the SNI-length field. (*ii*) `PAY`$_{abl}$ *ablates* the entire SNI from the CH. The removal shifts the next input bytes, potentially compensating the loss due to SNI removal by providing extra information compared to the original `PAY` input. (*iii*) `PAY`$_{pad}$ *pads* the SNI to 255 bytes reflecting the TLS 1.3 behaviour as described in the IETF draft[3], and *masks* it with random values. Unlike `PAY`$_{msk}$ and `PAY`$_{abl}$, this reduces useful real input, likely harming performance, as it may discard key information, such as server certificates and certification authority names (contained in the *Server Hello* (SH)), often crucial for distinguishing apps.

Meta Mimetic achieves an F1-score of 82.28% when no information removal scheme is applied (i.e., with the original `PAY` input). The different information removal schemes

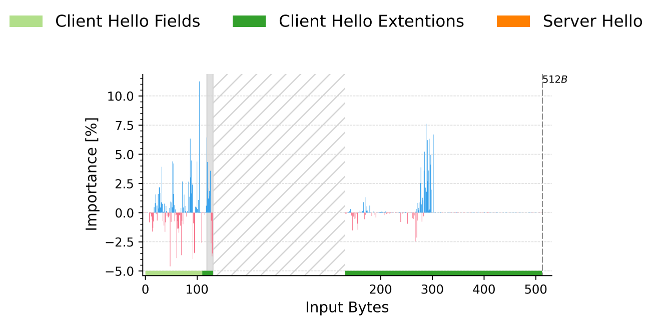
[3]https://datatracker.ietf.org/doc/draft-ietf-tls-esni/22/

produce varying effects on the performance. In particular, the $\texttt{PAY}_{abl}$ scheme is the least disruptive, causing only a negligible reduction in F1-score. When $\texttt{PAY}_{msk}$ is applied, the performance of META MIMETIC shows a small F1-score decrease of -0.62%. In contrast, $\texttt{PAY}_{pad}$ leads to a notable degradation: META MIMETIC has the largest reduction in F1 score (-3.49%). These results confirm the strong impact of the encryption of the SNI field when applied as defined in the IETF draft (i.w., with $\texttt{PAY}_{pad}$ scheme). When using other removal schemes ($\texttt{PAY}_{msk}$ and $\texttt{PAY}_{abl}$), the model retains similar performance by exploiting `PAY` portions still present in the SNI and bytes of the SH.

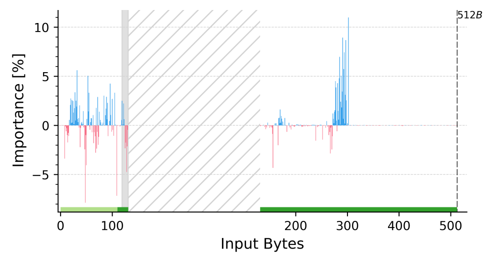
C. Exploring the Effects of Encryption on Byte-Level Feature Importance

To gain deeper insights and interpretability on the information removal schemes, we employ IG to assess importance scores to individual bytes in `PAY`. Figure 4 presents the input importance when `PAY` and its variants are fed to META MIMETIC, with positive and negative normalized IG values depicted with blue and pink colors, respectively.

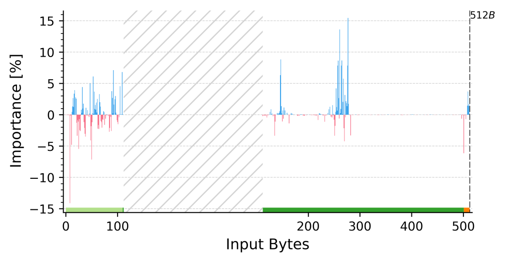
These inputs come from a `Zoom` biflow, where the CH ends at 516B but is truncated to $N_b = 512$B. The first 106B contain essential CH fields like the *Record Header*, *Handshake Header*, and *Cipher Suites*. Then, from 107^{th} byte to the end of the CH (407B in total), all CH extensions can be found, SNI included. Figure 4a shows the importance of using `PAY` as input. The shaded gray area highlights the portion of the input containing the server hostname, in this case, `www3.zoom.us`. A strong relative importance is observed in this region, peaking at 6%. The bytes before the domain consistently receive positive attribution, indicating that such bytes carry meaningful information for TC. In contrast, the final part, associated with the domain, shows negative importance, likely because it appears for multiple apps and therefore does not help the classifier in recognizing the `Zoom` app. The effect of $\texttt{PAY}_{abl}$ (shown in fig. 4c) is removing 14 bytes from the SNI extension, shifting the subsequent ones earlier in the payload. Since META MIMETIC leverages the first 512 bytes of the biflow, this shift causes some initial bytes of the SH (starting at byte 503) to fall within the analyzed range. META MIMETIC leverages some of these bytes for recognizing the app, as shown by their assigned importance. Fig. 4b illustrates the byte importance when masking the SNI value via $\texttt{PAY}_{msk}$. The overall attribution for the SNI extension becomes negative and the peak positive importance drops to 2.5%, while the negative importance reaches approximately -5%. Finally, Fig. 4d shows the disruptive impact of $\texttt{PAY}_{pad}$. As expected, the attributions for the bytes introduced by $\texttt{PAY}_{pad}$ are overall negligible, with roughly half receiving positive and half negative importance—confirming that these random inputs contribute little to the TC decision. Furthermore, adding 255 padding bytes not only removes the SNI but also partially excludes a region of high relative importance located near the 300^{th} byte of the original input.



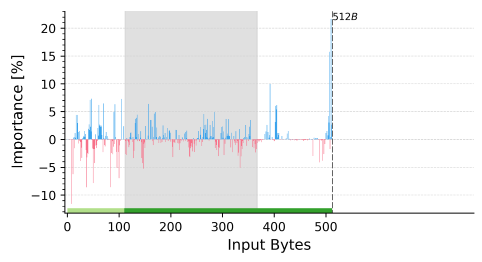
(a) `PAY`



(b) $\texttt{PAY}_{msk}$



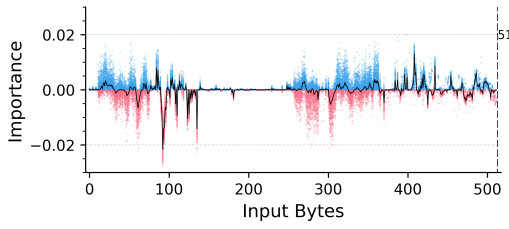
(c) $\texttt{PAY}_{abl}$



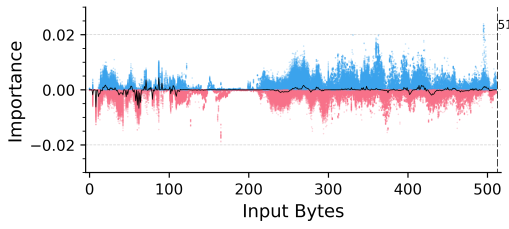
(d) $\texttt{PAY}_{pad}$

Fig. 4: Relative importance [%] for a `Zoom` biflow. **Gray areas** highlight: (a) the SNI server hostname (`www3.zoom.us`); (b) masked bytes of the hostname; (d) $255B$ padding replacing the SNI. The **dashed black line** marks $N_b = 512B$. **Gray hatched areas** in (a)–(c) aid visual comparisons.

Having examined how simulated encryption affects byte-level relevance in a `Zoom` biflow, we now shift our focus to real-world encrypted traffic. In particular, we analyze `Webex`, one of the apps with the highest proportion of TLS 1.3 biflows in `Mirage-2022`. This case allows us to observe how META MIMETIC responds to actual encryption scenarios and how TLS 1.3 reshapes feature importance. Fig. 5 compares importance scores for TLS 1.3 biflows and biflows with previous TLS versions within the `Webex` app. For completeness, in the



(a) TLS 1.3 biflows.



(b) non-TLS 1.3 biflows.

Fig. 5: Importance for TLS biflows of `Webex` biflow. The **continuous black line** is the median importance value.

plots, the median importance value of each byte (over different biflows) is reported as a solid black line. For TLS 1.3 biflows, a central region of the input (150–250 bytes) shows minimal importance, with consistently low median attributions. Beyond this range, more structured patterns emerge: the final bytes receive positive relevance, whereas bytes in the 100–150 range often assume negative importance. In contrast, non-TLS 1.3 biflows have a less defined importance profile, with no clearly influential regions, as the median importance remains negligible across most of the bytes, likely because relevant extensions (e.g., SNI) appear at varying positions, hindering the emergence of consistent patterns during explanation aggregation. These differences reveal how TLS 1.3 shapes classifier behavior, leading to distinct patterns in bytes importance.

VI. Conclusions and Future Perspectives

In this work, we investigated how encryption, particularly the adoption of TLS 1.3, affects the performance and interpretability of traffic classifiers. By focusing on the FSL-based Meta Mimetic classifier, we evaluated its behavior under increasing encryption schemes and real-world encrypted scenarios. In more detail, to assess the impact of encryption, we introduced controlled variants of the payload (`PAY`) input, progressively removing SNI-related bytes. Simulated encryption schemes using `PAY` input variants confirmed that the removal or masking of SNI-related data can degrade classification performance—especially in the case of more aggressive encryption strategies. Through Integrated Gradients, we observed that encryption reshapes the input focus of the classifier, shifting the importance away from previously relevant regions of the payload. The comparison between TLS 1.3 and non-TLS 1.3 biflows revealed distinct attribution patterns, highlighting how encryption not only reduces visibility but also alters the way models leverage the available input.

Future research perspectives include comparing multiple XAI techniques (e.g., SHAP, LRP, GradCAM) to gain a more comprehensive understanding of the classifier behavior under encryption. Another interesting, yet more challenging perspective is extending Meta Mimetic with continual learning capabilities to enhance its adaptability in dynamic network scenarios. Coupling continual learning with XAI analyses would improve adaptation and interpretability, providing insights into how feature importance evolves as the traffic landscape shifts.

References

[1] Statista, “Number of Smartphones Sold to End Users Worldwide from 2007 to 2023,” https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/, October 2024.
[2] Google, “HTTPS encryption on the web,” https://transparencyreport.google.com/https/overview?hl=en, October 2024.
[3] Fortinet, “The Challenges of Inspecting Encrypted Network Traffic,” https://www.fortinet.com/blog/industry-trends/keeping-up-with-performance-demands-of-encrypted-web-traffic/, August 2020.
[4] S. Rezaei, B. Kroencke, and X. Liu, “Large-Scale Mobile App Identification Using Deep Learning,” *IEEE Access*, vol. 8, pp. 348–362, 2019.
[5] C. Yang, Z. Gu, J. Bai, Z. Li, G. Xiong, G. Gou, S. Yao, and X. Chen, “Few-Shot Encrypted Traffic Classification: A Survey,” in *IEEE IPEC*, 2024, pp. 646–652.
[6] G. Bovenzi, D. Di Monda, A. Montieri, V. Persico, and A. Pescapè, “Meta Mimetic: Few-Shot Classification of Mobile-App Encrypted Traffic via Multimodal Meta-Learning,” in *ITC-35*, 2023, pp. 1–9.
[7] A. Nascita, G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapé, “A survey on explainable artificial intelligence for internet traffic classification and prediction, and intrusion detection,” *IEEE Communications Surveys & Tutorials*, 2024.
[8] Z. Zhang, P. Wang, T. Zhang, M. Liu, and X. Zhou, “Trustworthy generative few-shot learning based intrusion detection method in internet of things,” *IEEE Transactions on Consumer Electronics*, 2024.
[9] F. Cerasuolo, G. Bovenzi, V. Spadari, D. Ciuonzo, and A. Pescapè, “Explainable Few-Shot Class Incremental Learning for Mobile Network Traffic Classification,” in *IEEE GLOBECOM*, 2024, pp. 1299–1304.
[10] C. Yang, G. Xiong, Q. Zhang, J. Shi, G. Gou, Z. Li, and C. Liu, “Few-Shot Encrypted Traffic Classification via Multi-Task Representation Enhanced Meta-Learning,” *Computer Networks*, vol. 228, p. 109731, 2023.
[11] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, “Matching Networks for One-Shot Learning,” *Advances in Neural Information Processing Systems*, vol. 29, 2016.
[12] X. Ma, Y. Wang, Y. Lai, W. Jia, Z. Zhao, H. He, R. Yin, and Y. Chen, “A Multi-Perspective Feature Approach to Few-Shot Classification of IoT Traffic,” *IEEE TGCN*, vol. 7, no. 4, pp. 2052–2066, 2023.
[13] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, “MIMETIC: Mobile Encrypted Traffic Classification Using Multimodal Deep Learning,” *Computer Networks*, vol. 165, p. 106944, 2019.
[14] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller, and K. Hanssgen, “A Survey of Payload-Based Traffic Classification Approaches,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1135–1156, 2013.
[15] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, “Network Traffic Classifier with Convolutional and Recurrent Neural Networks for Internet of Things,” *IEEE Access*, vol. 5, pp. 18 042–18 050, 2017.
[16] M. Sundararajan, A. Taly, and Q. Yan, “Axiomatic Attribution for Deep Networks,” in *ICML*, 2017, pp. 3319–3328.
[17] “MIRAGE-2019 Dataset,” https://traffic.comics.unina.it/mirage/mirage-2019.html, 2019.
[18] “MIRAGE-COVID-CCMA-2022 Dataset,” https://traffic.comics.unina.it/mirage/mirage-covid-ccma-2022.html, 2022.
[19] G. Bovenzi, D. Di Monda, A. Montieri, V. Persico, and A. Pescapè, “Few Shot Learning Approaches for Classifying Rare Mobile-App Encrypted Traffic Samples,” in *IEEE INFOCOM WKSHPS*, 2023, pp. 1–6.
[20] A. Nascita, F. Cerasuolo, D. Di Monda, J. T. A. Garcia, A. Montieri, and A. Pescapè, “Machine and Deep Learning Approaches for IoT Attack Classification,” in *IEEE INFOCOM WKSHPS*, 2022, pp. 1–6.