PackMorph: Intelligent Traffic Signature Morphing for Privacy-Preserving Communications

Kunvar Kanhaiya†, Sourabh Singh†, Pralhad Magadum, and Aniruddha Singh Kushwaha
Department of Computer Science & Engineering Indian Institute of Technology Indore, India
(phd2201101015, phd2201201007, msrphd2304101012, phd2401101006, aniruddha)@iiti.ac.in

***Abstract*—Machine Learning(ML) models have enabled encrypted traffic classifiers to accurately identify services by analysing side-channel features such as packet sizes, timing behaviour, and burst dynamics, despite the payload itself remaining fully encrypted. These statistical fingerprints enable highly accurate monitoring of user activity, raising privacy concerns.**

To mitigate this threat, we present PackMorph, a cooperative traffic-shaping framework that counters encrypted-traffic classification. PackMorph employs a synchronised library of invertible statistical transformations to systematically adjust packet sizes, inter-arrival times, and flow volumes, thereby concealing the true identity of network flows and disrupting classifier accuracy.

Further empirical evaluation shows that PackMorph consistently forces target traffic to be misclassified as plausible decoy services, significantly reducing detection performance. Real-time emulation in Mininet further demonstrates the system's practical feasibility, achieving 100% lossless data reconstruction for complex workloads such as file transfers while preserving end-to-end protocol correctness. With only modest, policy-controlled impacts on latency and throughput, PackMorph provides a practical and deployable defense against sophisticated metadata-driven surveillance.

***Index Terms*—Privacy Preserving, User security, Encrypted Traffic Classification, Overlay Network, Side-Channel Fingerprint, Traffic Morphing.**

I. Introduction

Modern secure communication protocols such as TLS 1.3 [1], QUIC [2], and VPN tunneling [3] mechanisms have become the standard, ensuring that application-layer payloads remain unreadable to intermediate network entities. This widespread migration toward end-to-end encryption has effectively eliminated traditional deep packet inspection (DPI) techniques, marking a significant advancement in enhancing end-user confidentiality and privacy protection. As packet contents become increasingly inaccessible due to encryption, the focus has shifted to metadata-driven approaches to maintain visibility into network traffic. Consequently, efforts have been made to develop powerful encrypted-traffic classifiers capable of identifying applications and user activity without requiring payload decryption [4]. For instance, as illustrated in Fig. 1, a ML classifier exploits subtle side-channel traffic characteristics to identify the activity or application in use,

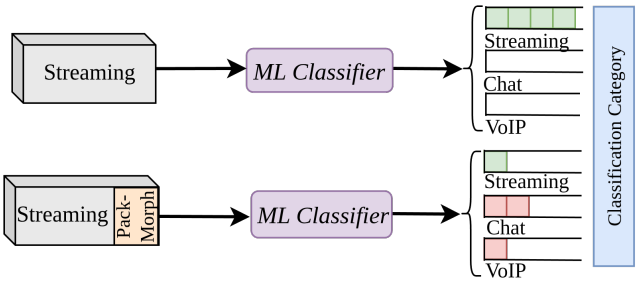


Fig. 1: Flow classification using ML classifier (Top) and introducing flow misclassification with PackMorph to introduce privacy (Bottom).

even in encrypted flows. Such characteristics include packet lengths, inter-arrival times, flow duration, directional burst structures, and the statistical distribution of packet sizes [5], [6]. The growth in computational resources, coupled with technological innovation, has accelerated the evolution of sophisticated ML classifier models. Such models range from ensemble methods like random forests to deep neural architectures such as convolutional neural networks (CNNs) [7]. These models are trained on flow-derived statistical attributes that demonstrate a high precision in recognizing encrypted network traffic [8].

The advances in traffic classification enhance several essential network functions. For example, they enable intrusion detection systems (IDS) to identify malware command-and-control communications [9]. They also support QoS enforcement [4] and facilitate lawful network management [10]. At the same time, however, they introduce significant privacy vulnerabilities. Despite full payload encryption, statistical side channels can still reveal fine-grained user activity patterns. An adversary can deduce behaviours such as video streaming, file transfers, VoIP sessions, or specific website access, thereby enabling surveillance, censorship, and profiling without breaking cryptographic protections. To counter this threat, a few countermeasures have been proposed to mitigate metadata leakage in encrypted communication. Early and common approaches rely on generic obfuscation, such as random padding to uniform lengths, flow tunnelling (e.g., onion routing), and packet-size normalisation [11]. While these methods obscure some basic statistics,

†Both authors contributed equally to this work.

ISBN 978-3-903176-82-9

others remain intact, such as timing periodicity and burst length. Furthermore, uncontrolled padding or random delay mechanisms frequently result in considerable bandwidth and latency overheads. Such a technique not only degrades the Quality-of-Experience (QoE) for end-users but also remains vulnerable to advanced classifiers that are specifically trained on noise-augmented datasets. Crucially, most existing techniques operate unilaterally at the sender without receiver cooperation. This architectural limitation prevents aggressive shaping because the receiver lacks the context to reconstruct the stream, leading to TCP retransmission timeouts or application failure.

These limitations highlight two major, unaddressed gaps in the current landscape of privacy-enhancing technologies. First, the lack of synchronised transformation semantics between sender and receiver which, prevents the system from performing complex modifications without compromising the receiver's reassembly process or triggering TCP timeouts. Second, existing mechanisms often rely on generic noise, which is bandwidth-inefficient and detectable. Therefore, an effective privacy framework requires a targeted mechanism that replaces a flow's identifiable traits with the statistical distribution of a plausible alternative application. To address these challenges, we propose a novel framework, *PackMorph* an AI-enabled intelligent network traffic morphing system that reshapes encrypted traffic to statistically resemble a different plausible set of service classes, as illustrated in Fig. 1 (bottom). The contribution of this work is as follows:

1) **Design of PackMorph:** We design PackMorph, a cooperative tunnelling framework for reversible traffic shaping. It incorporates a presynchronized signalling mechanism, in which transformation rules are securely transferred before data transmission for a flow.
2) **Empirical Evaluation:** We evaluated PackMorph against Random Forest classifiers using the MIT-VNAT dataset. The results demonstrate that PackMorph effectively neutralizes traffic analysis, reducing the classifier's service identification accuracy and successfully forcing the missclassification of target flow into alternate service categories.
3) **Real-time Validation:** We validate the practical deployment of PackMorph through a real-time Mininet emulator. The experiments confirm that the system maintains lossless data reconstruction and protocol correctness.

Functionally, PackMorph decouples the semantic content of a flow from its statistical presentation. By selectively manipulating side-channel features, specifically packet sizing, inter-arrival timing, and burst dynamics, the system effectively masks the original traffic signature to align with a target service profile, all while treating the encrypted payload as an opaque, untouched stream. To guarantee reliability, PackMorph relies on deterministic state synchronisation between sender and receiver. This ensures that every shaping operation is perfectly reversible, allowing the receiver to extract the original traffic without error.

This paper is organised as follows: Section II reviews related work in traffic analysis evasion and mimicry. Section III introduces the overall design of PackMorph and its operational workflow. Section IV presents the experimental setup and evaluation results, covering both privacy effectiveness and operational performance. Section V provides a discussion on the trade-offs and practical implications of the system. Finally, Section VI concludes the paper.

II. Related Work

The field of encrypted traffic privacy is defined by an ongoing conflict between traffic classification techniques and the countermeasures developed to defeat them. Extensive research addresses encrypted traffic classification, wherein encryption protects payloads, yet ML techniques can infer sensitive details from flow metadata, including packet sizes, timing, and burst characteristics [4], [12]. This line of research has progressed from classical ML models, such as Random Forests, which rely on handcrafted statistical features, to more powerful Deep Learning architectures, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), capable of learning representations directly from raw packet sequences [7], [13], [14]. More recently, the paradigm has shifted towards Transformer-based foundation models and self-attention mechanisms, which excel at capturing long-range spatiotemporal dependencies across hierarchical traffic levels without relying on rigid byte truncation [15], [16]. Such ML classifiers have demonstrated high effectiveness in real-world applications, including website fingerprinting [5], [17], where an adversary can infer the specific website a user is accessing, even when traffic is routed through the Tor anonymity network. Prior work has successfully distinguished Tor from non-Tor traffic [8] and identified specific user activities within VPN tunnels [18]. A key limitation exposed by this work is that encryption safeguards payloads but leaves metadata unprotected, resulting in exploitable side channels and motivating the development of corresponding defense strategies..

To mitigate these side-channel leaks, defense strategies have evolved from simple heuristic obfuscation to complex generative approaches. Early and common approaches rely on generic obfuscation, such as random padding to uniform lengths, flow tunnelling, and packet-size normalization [19], while others attempt to reshape traffic to a Constant Bit Rate (CBR) to conceal bandwidth usage signatures [20]. However, these techniques frequently impose substantial bandwidth and latency overheads, significantly impairing the user's QoS. Furthermore, they generally exhibit poor resilience to adaptive attacks; an adversary can retrain the classifier on obfuscated data and often re-identify modified patterns with accuracy comparable to the original model [6]. Recognizing these limitations, recent research has pivoted toward advanced countermeasures leveraging Generative Adversarial Networks (GANs) to synthesize traffic-padding patterns that mimic benign flows [21]. These generative methods represent a significant advancement over random padding by

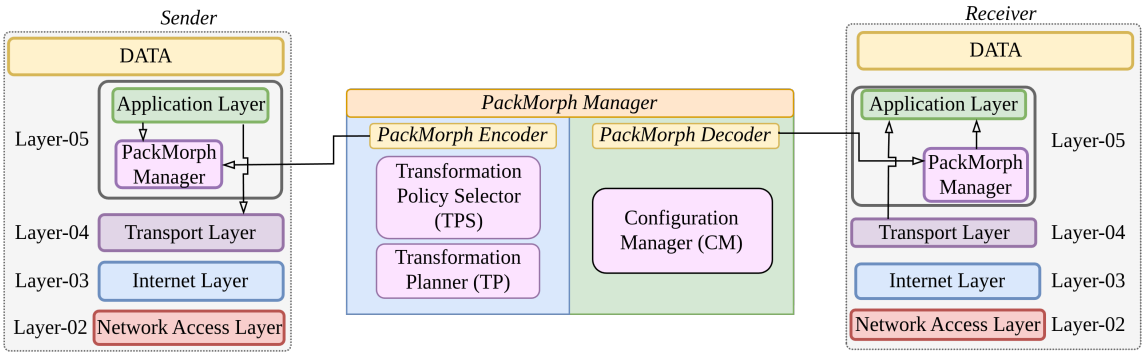


Fig. 2: High-level architecture of the PackMorph Manager.

attempting to learn and replicate realistic flow dynamics for real-world deployment. The prohibitive computational cost of performing deep learning inference per packet introduces significant latency, creating a bottleneck that high-speed network infrastructure cannot sustain.

Crucially, regardless of their generation strategy, most existing techniques suffer from a critical architectural blindness: they operate unilaterally at the sender without receiver cooperation [22], [23]. This lack of synchronization creates a fundamental reliability bottleneck, as the receiver possesses no context to distinguish legitimate traffic shaping from network anomalies. Consequently, unilateral defenses are prevented from performing aggressive shaping; any significant deviation in packet sizing or timing, such as coalescing packets or introducing substantial delays, risks violating the receiver's TCP state machine. In the absence of a cooperative restoration mechanism, these modifications frequently trigger spurious retransmission timeouts, disrupt stream reassembly, and ultimately lead to application failure.

Our proposed PackMorph addresses the core metadata visibility problem. It fundamentally reshapes a flow's traffic parameters to mimic those of an entirely different flow class. This class-to-class transformation neutralizes adaptive attacks by forcing a retrained classifier to learn an incorrect mimicked mapping, all while maintaining strict protocol synchronization between the sender and the receiver.

III. System Overview

In this section, we present the system framework, detailing the architectural design and its components. We also explain the system's operational workflow, detailing the underlying algorithms and synchronisation mechanisms that helps hiding the traffic semantics.

A. System Framework

The proposed framework is a cooperative end-to-end flow obfuscation mechanism designed to defend against ML–based traffic analysis and fingerprinting. The PackMorph framework adopts a symmetric architecture deployed across both communication endpoints (PackMorph Manager), as illustrated in Fig. 2. It operates on creating an end-to-end "Traffic Morphing Tunnel", where the source and destination endpoints cooperate to reshape traffic according to a negotiated statistical profile.

1) PackMorph Manager: The PackMorph manager is composed of two primary functional modules that manage the lifecycle of a flow: the encoder at the sender and the decoder at the receiver.

The encoder is responsible for the active transformation of traffic. It intercepts the original payload and restructures it to match a target statistical distribution. To realize this functionality, the framework coordinates three internal submodules. The *Traffic Profiler & Interceptor* captures outbound packets from the virtual interface, classifies each flow, and buffers the payload, thereby decoupling application data from its original temporal characteristics. The *Transformation Policy Selector* (TPS) then serves as the decision engine, consulting an offline morphing library (see Sec. III-B) to determine an appropriate target service class. Based on this selection, it formulates a high-level transformation policy that specifies the required statistical adjustments to packet sizes, inter-arrival times, and overall flow volume. The selected transformation policy needs to be communicated to the Configuration Manager(CM) of the PackMorph at the receiver end for synchronisation. Finally, the *Transformation Planner & Executor* (TP) converts this policy into concrete packet-level actions. The TP carries out the physical transformation by following these steps: 1) It performs controlled packet fragmentation and padding to satisfy packet size constraints. 2) It injects dummy packets to meet aggregate flow volume targets. 3) It enforces precise timestamp scheduling to conform to the desired inter-arrival profile. Finally, it forwards the reshaped traffic to the physical transport layer.

At the receiving end, a decoder is used to remove the artificial traffic shapes and recover the original application data with complete accuracy. The restoration is achieved in three steps: 1) It processes the initial packet of a transmission epoch to extract the action vector, which tells the system

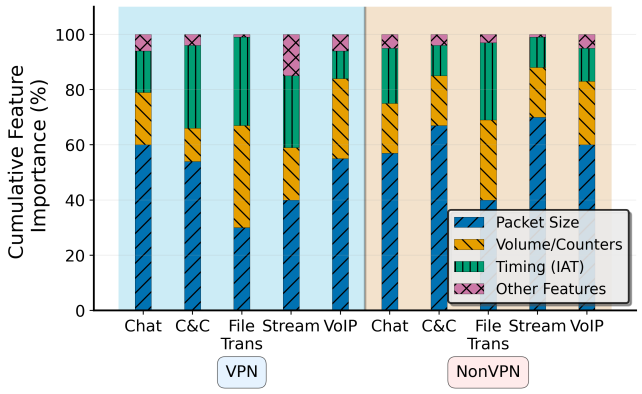


Fig. 3: Empirical feature importance and 3D transformation vector space utilised by the PackMorph Manager.

exactly how the incoming traffic was modified. 2) Using the action vector, it discards injected dummy packets and strips away padding to isolate the genuine payload bytes. 3) The fragments gathered to rebuild the original application messages and deliver the unshaped data to the receiving application through the virtual interface.

2) Architecture & Layer Abstraction: To ensure broad compatibility and address the "transparency" challenges often associated with traffic shaping, PackMorph is implemented as a userspace shim layer residing logically between the Application Layer and the Transport Layer. The practical implementation is achieved via virtual network interface(e.g., TUN/TAP devices). This architectural choice is critical for two reasons: First, it ensures strict application transparency. Because applications bind to the virtual interface rather than the physical Network Interface Controller (NIC), the system intercepts distinct flows before they enter the kernel's networking stack. Consequently, the application remains completely agnostic to the underlying shaping and continues its standard socket operations without modification. Second, this design provides isolation from network dynamics. By buffering packets at the shim layer, PackMorph isolates the application from the latency and jitter introduced by the shaping process. The local manager handles the buffering and scheduling compliance, while the standard TCP/IP stack ensures reliable delivery over the physical link. This operational isolation mitigates the risk of triggering application-layer timeouts or retransmission storms

B. System Working

The operational workflow of PackMorph is divided into two distinct phases: an offline profile construction phase and an online real-time execution phase. This separation allows the system to remain computationally lightweight during active transmission.

1) Phase I: Offline Profile Construction: The effectiveness of the morphing process depends critically on the accuracy of the underlying statistical models. Although conventional flow extraction tools (e.g., CICFlowMeter [9]) compute more than 80 statistical features per flow, leveraging the full feature set incurs substantial computational overhead, making it impractical for real-time data-plane deployment. To ensure the system remains lightweight while maintaining high obfuscation accuracy, the TPS employs a reduced-dimensionality approach. As shown in Fig. 3, empirical evaluation of feature importance across diverse VPN and non-VPN traffic classes shows that three feature groups–packet length, byte and packet counters, and inter-arrival time–collectively account for an average of 87.5% of the discriminatory variance leveraged by advanced classifiers. Exploiting these three dominating features, the TPS constructs a library of morphing profiles, denoted as $\mathcal{L} = \{\boldsymbol{\mu}_c \mid c \in \mathcal{C}\}$, where $\mathcal{C}$ represents the set of supported service classes (e.g., video streaming, VoIP, file transfer, etc.). Each profile $\boldsymbol{\mu}_c$ serves as a statistical "fingerprint" for class c, defined by a vector of learned distribution parameters:

$$\boldsymbol{\mu}_c = [\boldsymbol{\mu}_{c,\text{size}},\ \boldsymbol{\mu}_{c,\text{iat}},\ \boldsymbol{\mu}_{c,\text{vol}}]$$

where $\boldsymbol{\mu}_{c,\text{size}}$ models the spatial packet size distribution, $\boldsymbol{\mu}_{c,\text{iat}}$ models the temporal inter-arrival time (IAT) distribution, and $\boldsymbol{\mu}_{c,\text{vol}}$ captures the volumetric flow characteristics. These profiles are built offline and pre-loaded into PackMorph instances, ensuring no heavy training overhead is incurred during live transmission.

2) Phase II: Real Time Execution: During active data transmission, the system executes a cyclic process of "Plan-Act-Verify" as shown in Fig.4. The *plan* process derives the transformation rules for a given epoch and communicates them to the receiver. The *act* process then enforces these rules to morph the traffic accordingly. The *verify* process monitors the transition to a new epoch and triggers a return to the plan phase. All three processes are orchestrated by the Transformation Planner (TP), which operates in discrete time epochs (t).

Strategy Generation: When a new flow of source class c_{src} is detected, the TPS selects a target morphing class c_{tgt} such that $c_{\text{tgt}} \neq c_{\text{src}}$. It then computes a shift vector, Δ, which quantifies the statistical modifications required across all three target dimensions:

$$\Delta = [\Delta_{\text{size}}, \Delta_{\text{iat}}, \Delta_{\text{vol}}]$$

To prevent long-term statistical drift that advanced classifiers could detect, the TPS periodically refreshes c_{tgt}, triggering a dynamic update of the transformation vector. Thereby changing the morphing strategy in every epoch.

Packet-Level Planning: The TP translates the abstract goals of Δ into a concrete, epoch-based execution schedule. For a given epoch window, the TP calculates the strict volume of packets required (N_{target}), the required packet sizes (S_{target}), and the precise transmission timestamps (T_{target}). By comparing these targets against the actual buffered application data (N_{actual}), the TP generates an action vector $\boldsymbol{a}_{\text{epoch}}$ containing three specific rule sets:

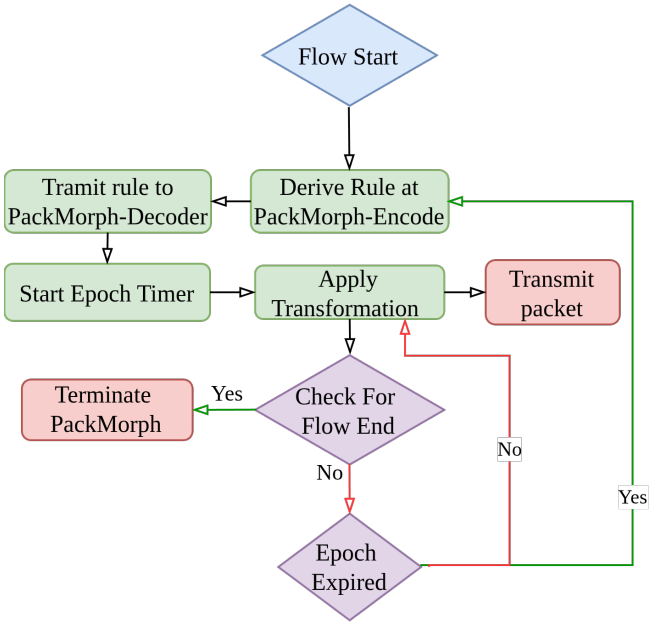


Fig. 4: PackMorph Transformation Workflow.

- **Sizing Rule** (P_{rule}) dictates the fragmentation of application data if it exceeds S_{target}, or the addition of cryptographic padding if it falls short.
- **Timing Rule** (T_{rule}) enforces a strict release schedule, introducing micro-delays to perfectly match the burst transmission patterns of c_{tgt}.
- **Volume Rule** (V_{rule}) manages packet counts. In an underflow scenario ($N_{\text{actual}} < N_{\text{target}}$), the system generates $N_{\text{dummy}} = N_{\text{target}} - N_{\text{actual}}$ packets. Conversely, in an overflow scenario ($N_{\text{actual}} > N_{\text{target}}$), the system enforces a queue threshold by buffering the excess packets for the subsequent epoch, preventing statistical volume spikes.

Synchronisation and Restoration: A critical requirement of PackMorph is that the receiver must accurately reverse the transformation without maintaining a complex, parallel state machine. To achieve this via in-band synchronization, the configuration metadata for the current epoch, specifically the map of dummy packets and padding lengths defined by $\boldsymbol{a}_{\text{epoch}}$, is encoded into a Synchronization Header. This encrypted header is embedded within the first packet of the epoch. Upon receipt, the remote PackMorph-Restore component decodes this header and applies the inverse rules. It filters the stream, silently dropping all packets flagged as dummies by V_{rule}, and reassembles the remaining payloads by stripping the padding defined by P_{rule}. Ultimately, this ensures that while a network observer perceives the complex statistical patterns of the target class, the end user application receives the original data stream with perfect fidelity.

TABLE I: Applications and Traffic Categories

ID	Traffic Category	Applications
S0	Streaming	Vimeo, Netflix, YouTube
S1	VoIP	Zoiper
S2	Chat	Skype
S3	Command & Control	SSH, RDP
S4	File Transfer	SFTP, RSYNC, SCP

IV. Simulation And Results

In this section, we validate the practical effectiveness of the proposed PackMorph. We conducted both dataset-driven and real-network-driven experiments. Our objective is twofold:

- To quantify the effectiveness of the morphing by analysing the change in classification of a target traffic classifier when processing the transformed encrypted traffic.
- To verify operational transparency by confirming that all applied transformations are fully and symmetrically reversed at the receiver, thereby maintaining end-to-end application integrity.

The following subsections describe the evaluation methodology, benchmarking and experimental emulation results.

A. Methodology

1) Dataset and Feature Extraction: We utilize the VPN/Non-VPN Network Application Traffic Dataset (VNAT) released by MIT Lincoln Laboratory [24], which contains encrypted VPN-wrapped and non-VPN packet captures across multiple service categories. To ensure a comprehensive evaluation, we focus on the five distinct traffic categories present in the dataset for both VPN/non-VPN, detailed in Table I, along with their representative applications.

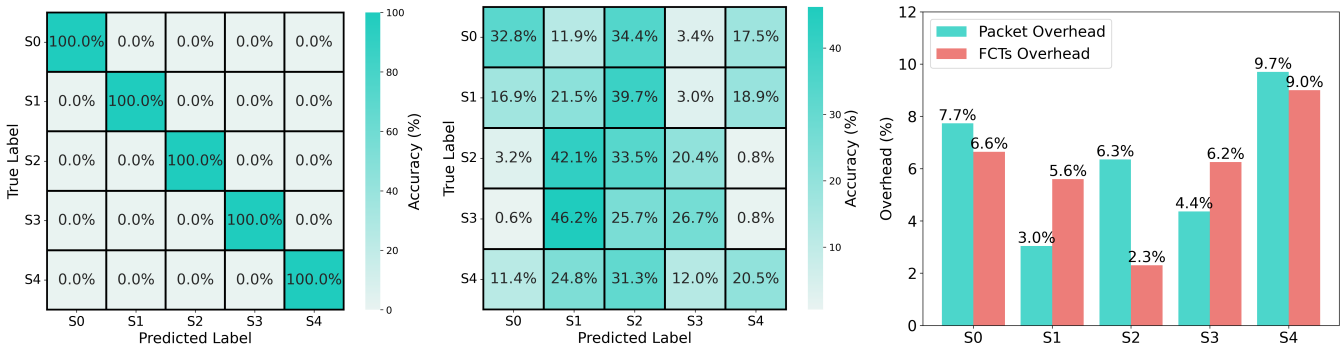
All PCAP traces in [24] are converted into bidirectional flow records using *CICFlowMeter* [9], which aggregates more than 80 traffic features widely adopted in encrypted traffic analysis. These flow statistics serve as the input representation for our classifier-based evaluation.

2) Transformation Policy Derivation: Statistical templates are derived for each service category through an offline analysis of smaller PCAP subsets. These templates capture realistic distributions of packet sizing ($\boldsymbol{\mu}_{\text{size}}$), inter-arrival pacing ($\boldsymbol{\mu}_{\text{iat}}$), and volumetric burst structures ($\boldsymbol{\mu}_{\text{vol}}$). The TPS then utilizes these learned templates as morphed targets for real-time transformation.

3) Classifier Model Training: A Random Forest (RF) classifier is used as the adversarial prediction model, following prior encrypted-traffic classification studies. We apply an 80/20 stratified train-test split to preserve class balance. The RF model consists of 100 trees with a maximum depth of 20, and incorporates automatic class balancing to mitigate skew in traffic category distribution.

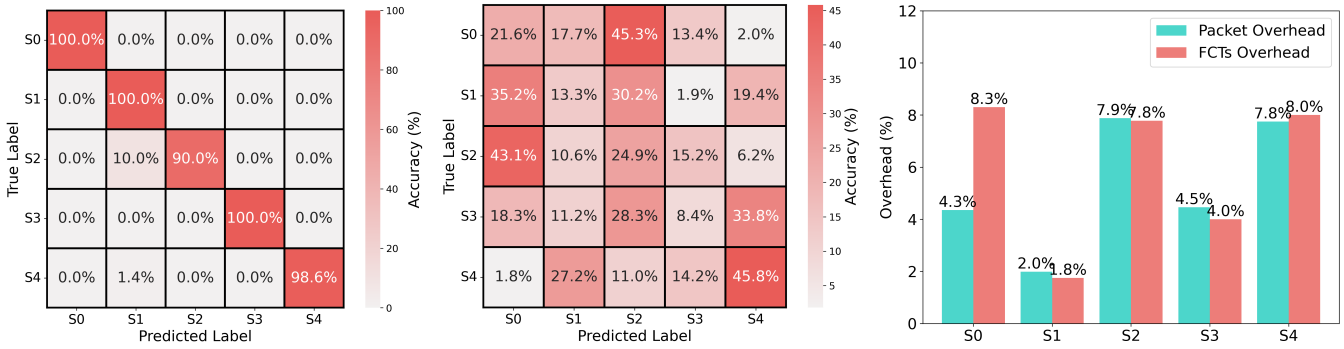
B. Benchmarking

For benchmarking the PackMorph framework, we investigate how effectively PackMorph suppresses encrypted-traffic



(a) Original Non-VPN Accuracy (b) Morphed Non-VPN Accuracy (c) Non-VPN Performance Overhead

Fig. 5: Analysis of PackMorph performance for Non-VPN traffic. (a) Classifier confusion matrix for original traffic, (b) Confusion matrix after PackMorph transformation, and (c) Performance overhead in terms of bandwidth and flow duration for each service.



(a) Original VPN Accuracy (b) Morphed VPN Accuracy (c) VPN Performance Overhead.

Fig. 6: Analysis of PackMorph performance for VPN traffic. (a) Classifier confusion matrix for original traffic, (b) Confusion matrix after PackMorph transformation, and (c) Performance overhead in terms of bandwidth and flow duration for each service.

classification prediction in both VPN and non-VPN environments.

1) Dataset-Based Evaluation: Figure 5a and 6a present the baseline RF classification results for VPN and non-VPN flows, respectively. The classifier demonstrates strong recognition performance with high per-class prediction, confirming that flow-based encrypted-traffic analysis remains effective.

After applying PackMorph transformation, the classifier is executed again on the transformed flows. Fig. 5b and 6b show a clear performance degradation as classifications are redistributed towards the other services rather than the true service identities (represented by diagonal fields). This verifies that PackMorph successfully masks statistical fingerprints. Importantly, the degradation trend is consistent across both VPN and non-VPN settings, indicating that PackMorph is robust to underlying encryption methods. We now evaluate the cost of our system by measuring the overhead.

a) Packet Overhead: The packet overhead is introduced in transformations by adding padding bits or fragmentation. The packet overhead is calculated as the percentage increase in the total number of bytes transmitted. An average increase in packet overhead of 3.144% and 4.35% is observed for VPN and non-VPN services, respectively.

b) Flow Completion Time (FCT) Overhead: A critical Quality-of-Experience (QoE) metric is the flow completion time. Due to packet overhead, flow completion time also increases. We measure this overhead by comparing the durations from the first to the last packet in the original flows with those in their transformed counterparts. An average increase of 6.5% and 6.16% is observed for VPN and non-VPN services, respectively.

C. Experimental Emulation

To evaluate the operational feasibility of the proposed framework, we deployed PackMorph within a Mininet-based emulation environment, as shown in Fig. 7. The setup consists of two virtual hosts connected via an Open vSwitch (OVS). The PackMorph encoder module operates on the sender side and is responsible for generating and applying the transformation plan. During the initial transmission, the metadata associated with this plan is transmitted to the receiver's PackMorph decoder unit. The decoder then uses

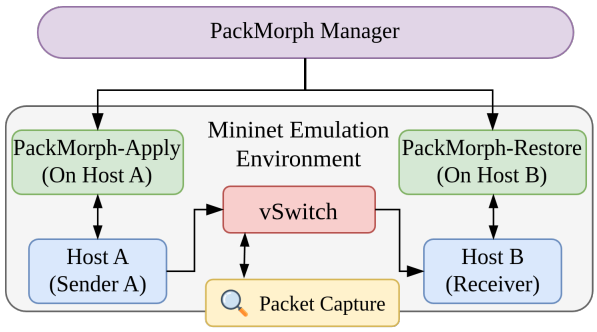


Fig. 7: PackMorph Emulation Setup

this information to reverse the applied transformations and reconstruct the original flow characteristics in a lossless manner.

The experimental testbed was hosted on an Ubuntu 24.04 LTS machine featuring an Intel Core i9-14900K processor, 128 GB of DDR5 RAM, and a 2 TB NVMe SSD. We emulated the network topology using Mininet and Open vSwitch, setting a baseline link capacity of 250 Mbps. The core transformation and restoration engines were implemented in Python (v3.10) to manage real-time packet manipulation, flow parsing, and metadata synchronization. To evaluate the system, we routed both synthetic and replayed real-world traffic traces through this emulated network. The modified traffic was then captured using *Wireshark* to calculate key performance metrics, specifically Evasion Success, Bandwidth Overhead, and FCT Impact.

1) Emulation Results: For the evaluation, we set the file transfer flow, as their high payload variability and throughput sensitivity make them an ideal and challenging workload for assessing both transformation overhead and restoration fidelity under real-time operation. The evaluation results for file transfer flows demonstrate that PackMorph preserves end-to-end correctness while introducing manageable communication overhead. Key observations include:

- Restoration Accuracy: The PackMorph-decoder achieves 100% bit-exact reconstruction of all transmitted packets using the CM.
- Bandwidth Overhead: A moderate increase of 11.54–11.64% in transmitted bytes is observed, primarily due to padding and dummy packet insertion.
- Delay Impact: Transformation-induced pacing introduces an average flow Completion time (FCT) by 12.87% compared to the baseline transmission.
- Protocol Transparency: No TCP retransmissions, flow resets, or application-visible performance degradation are detected throughout the evaluation.

Moreover, when the post-transformation file transfer traffic is re-evaluated using the trained classifier, it is predominantly identified as a non-file-transfer service. This outcome confirms that the morphing objective remains effective even during live transmission. We analysed overhead across different file sizes; the measured overhead varies gradually, as shown in Table II, indicating the scalability with flow volume.

2) Comparison with Existing Schemes: To contextualize PackMorph's performance, we compared it against three established traffic obfuscation frameworks: Obfsproxy (Scramblesuit) [25], Marionette [26], and AdvTraffic [27]. We evaluated these schemes based on their ability to deceive the same Random Forest Classifier and their associated bandwidth/latency costs.

- Evasion Effectiveness: While Obfsproxy [25] provides low-latency randomization, it only achieves a ∼15% reduction in classification accuracy because it does not transform specific application shapes. In contrast, PackMorph achieves accuracy reduction of over 70%, delivering evasion performance comparable to the GAN-based AdvTraffic [27], but with significantly lower computational complexity.
- Bandwidth/Latency Trade-Off: Existing high-evasion schemes often incur steep performance penalties, AdvTraffic [27] suffers from bandwidth bloat exceeding ∼40%, while Marionette [26] introduces a ∼40% latency penalty due to complex state-machine processing. PackMorph balances this trade-off by using pre-derived statistical templates, maintaining a low bandwidth overhead of ∼11% and limiting latency to approximately ∼12%, thereby offering superior operational efficiency compared to the contemporary works.

TABLE II: Performance Overhead of File Transfer Flows with PackMorph

Flow ID	Size (MB)	Overhead (%)	Delay (%)
F1	5	11.54	12.85
F2	25	11.64	12.89
F3	50	11.63	12.86
F4	100	11.64	12.86

TABLE III: Comparative Analysis of PackMorph vs. Existing Schemes for 100MB File Transfer

Scheme	Mimicry Method	Evasion Success	Bandwidth Overhead	FCTs Impact
Obfsproxy [25]	Uniform Randomization	Low (∼15.3%)	Very Low (<2%)	Low (∼15%)
Marionette [26]	Grammar-based	High (∼33.7%)	Moderate (∼18%)	High (∼40%)
AdvTraffic [27]	GAN-based	High (∼59.2%)	High (∼40%)	Moderate (∼32%)
PackMorph (Ours)	**Statistical Templates**	**Very High (∼89.8%)**	**Low-Moderate (∼11%)**	**Low (∼12%)**

V. DISCUSSION

A. Performance and Privacy Trade-offs

While PackMorph provides robust privacy guarantees, traffic morphing inherently introduces operational overhead. Forcing a low-volume flow (e.g., SSH or Chat) to be transformed to a high-volume flow (e.g., Video Streaming) requires the injection of dummy packets, directly impacting bandwidth efficiency. Conversely, manipulating inter-arrival times (IATs) to match a target burst profile inevitably introduces artificial latency, potentially degrading the Quality of Experience (QoE) for latency-sensitive applications. However, because PackMorph allows selective, policy-driven transformation where target classes can be chosen based on their statistical proximity to the source flow, the system can dynamically optimize this trade-off, minimizing overhead while maximizing classifier confusion.

B. Limitations and Future Work

While the current version of PackMorph proves highly resilient against classical ensemble classifiers such as Random Forests, the threat landscape of encrypted traffic analysis is continuously evolving. Future evaluations must benchmark the framework against advanced Deep Learning architectures, specifically Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), which excel at extracting complex, non-linear spatial and temporal dependencies directly from raw packet sequences. Furthermore, the current prototype relies on an offline profile construction phase, which assumes relatively stable network conditions. In real-world, highly congested networks, natural jitter and packet loss can unintentionally distort the carefully crafted IATs, potentially reducing the morphing fidelity. To counter this, future work will focus on integrating dynamic, on-the-fly profile generation and exploring the offloading of the Transformation Planner (TP) logic to programmable data planes to process traffic at line rate, significantly reducing the base latency of the shim layer.

VI. CONCLUSION

The widespread adoption of end-to-end encryption has successfully secured application payloads but has inadvertently elevated the importance of metadata, which enables adversaries to profile user behavior through sophisticated ML classifiers. To address this critical vulnerability, this paper introduced PackMorph, a novel, cooperative traffic-shaping framework designed to counter encrypted-traffic classification. By leveraging a synchronized library of invertible statistical transformations, PackMorph strategically manipulates packet sizes, inter-arrival times, and flow volumes to mask the true identity of network streams, rendering conventional classifiers ineffective. Our empirical evaluations demonstrate that the system successfully misclassifies target traffic as plausible decoy services, substantially reducing an adversary's detection accuracy. Furthermore, real-time Mininet emulation validated the framework's practical viability, confirming 100% lossless data recovery for complex streams, such as file transfers, without compromising end-to-end protocol reliability. By introducing only modest, policy-controlled overheads in latency and throughput, PackMorph establishes a robust, highly deployable defence mechanism for preserving user privacy against advanced metadata surveillance.

REFERENCES

[1] E. Rescorla, "The transport layer security (tls) protocol version 1.3," Tech. Rep., 2018.

[2] J. Iyengar and M. Thomson, "Rfc 9000: Quic: A udp-based multiplexed and secure transport," *Omtermet Emgomeeromg Task Force*, p. 4, 2021.

[3] S. Kent and K. Seo, "Security architecture for the internet protocol," Tech. Rep., 2005.

[4] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE communications surveys & tutorials*, vol. 10, no. 4, pp. 56–76, 2009.

[5] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier," in *Proceedings of the 2009 ACM workshop on Cloud computing security*, 2009, pp. 31–42.

[6] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 1928–1943.

[7] M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.

[8] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related," in *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, 2016, pp. 407–414.

[9] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, no. 2018, pp. 108–116, 2018.

[10] ETSI, "Lawful Interception (LI); Handover specification for IP delivery," European Telecommunications Standards Institute, Tech. Rep. TS 102 232, May 2006.

[11] D. A. Worae and S. Mastorakis, "Hiding in plain sight: An iot traffic camouflage framework for enhanced privacy," *arXiv preprint arXiv:2501.15395*, 2025.

[12] P. Velan, M. Čermák, P. Čeleda, and M. Drašar, "A survey of methods for encrypted traffic classification and analysis," *International Journal of Network Management*, vol. 25, no. 5, pp. 355–374, 2015.

[13] W. Wang, M. Zhu, X. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2017, pp. 43–48.

[14] 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.

[15] J. Zhang, H. Zhao, Y. Feng, Z. Cai, and L. Zhu, "Netst: Network encrypted traffic classification based on swin transformer." *Computers, Materials & Continua*, vol. 84, no. 3, 2025.

[16] S. Mayhoub, C. H. Foh, M. B. Mashhadi, M. Shojafar, and R. Tafazolli, "Talk like a packet: Rethinking network traffic analysis with transformer foundation models," *arXiv preprint arXiv:2602.06636*, 2026.

[17] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at scale." in *NDSS*, vol. 16, 2016, pp. 1–15.

[18] G. D. Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related features," *ICISSP 2016*, 2016.

[19] C. V. Wright, S. E. Coull, and F. Monrose, "Traffic morphing: An efficient defense against statistical traffic analysis," in *NDSS*, vol. 9, 2009, pp. 237–250.

[20] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 143–157.

[21] M. Ring, D. Schlögel, D. Landes, and A. Hotho, "Flow-based network traffic generation using generative adversarial networks," *Computers & Security*, vol. 82, pp. 156–172, 2019.

[22] T. Wang and I. Goldberg, "Walkie-talkie: An efficient defense against passive website fingerprinting attacks," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 1375–1390.

[23] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an efficient website fingerprinting defense," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 1, pp. 27–46, 2016.

[24] S. Jorgensen, J. Holodnak, J. Dempsey, K. de Souza, A. Raghunath, V. Rivet, N. DeMoes, A. Alejos, and A. Wollaber, "Extensible machine learning for encrypted network traffic application labeling via uncertainty quantification," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 1, pp. 420–433, 2023.

[25] P. Winter, T. Pulls, and J. Fuss, "Scramblesuit: A polymorphic network protocol to circumvent censorship," in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, 2013, pp. 213–224.

[26] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Marionette: A programmable network traffic obfuscation system," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 367–382.

[27] H. Liu, J. Dani, H. Yu, W. Sun, and B. Wang, "Advtraffic: Obfuscating encrypted traffic with adversarial examples," in *2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*, 2022, pp. 1–10.