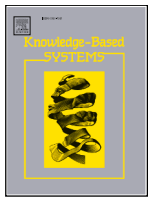
Knowledge-Based Systems 342 (2026) 115893

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



# Unveiling user activities on instant messaging platforms: A study of activity fingerprinting through traffic analysis and machine learning techniques

Lorena Mehavilla *, José García , Álvaro Alesanco

*Aragón Institute of Engineering Research (I3A), University of Zaragoza, C/ Mariano Esquillor Gómez, Zaragoza, 50018, Zaragoza, Spain*

ARTICLE INFO

*Keywords:*
Instant message forensic
Traffic characterisation
Network traffic analysis
Machine learning
Activity fingerprinting

ABSTRACT

Encrypted instant messaging (IM) traffic conceals message content but still exposes communication patterns that can reveal user behaviour. This paper presents a unified framework for inferring user activities across multiple IM platforms by analysing encrypted traffic using machine learning techniques. The proposed approach integrates empirical traffic characterisation, transaction-centric segmentation, and lightweight classifiers to detect user actions, such as sending or receiving text and multimedia messages, in real time. Using Zeek as the core analysis engine, the framework performs packet inspection, transaction segmentation, connection classification, and feature extraction. The framework was evaluated on traffic from nine major IM platforms (Discord, Facebook Messenger, Instagram, Snapchat, Microsoft Teams, Telegram, WeChat, WhatsApp, and X), achieving F1 scores ranging from 0.62 for X up to 0.98 for WhatsApp. Unlike prior studies limited to single applications or synthetic datasets, our work employs realistic, user-driven traffic and explicitly distinguishes message type and direction, improving comparison and cross-platform generalization. Beyond methodological advancements, this study exposes privacy risks inherent in encrypted communication and outlines ethical safeguards and countermeasures to mitigate activity fingerprinting. The findings demonstrate that accurate, real-time inference of encrypted messaging activities is feasible under responsible, consent-based conditions, offering valuable insights for network forensics and privacy-aware communication design.

## 1. Introduction

Instant messaging (IM) platforms have become integral to modern communication, enabling real-time interactions across geographical boundaries and time zones. Their widespread adoption underscores their importance in both personal and professional settings, making them a critical focus for understanding contemporary social dynamics and the technological challenges associated with secure communication.

Instant messaging applications such as WhatsApp, WeChat, Facebook Messenger, Telegram and Snapchat are among the most widely used communication tools today [1,2]. WhatsApp leads the rankings with approximately 2 billion active users worldwide, followed by WeChat and Facebook Messenger, each exceeding one billion users. Although social networks are not primarily designed for instant messaging, many, such as Instagram, Discord, X, and Teams, have integrated messaging services due to their popularity and extensive use.

These applications support a wide range of user actions, including text messaging, voice and video calls, file transfers, and signalling activities such as typing notifications. To protect user privacy, IM platforms employ various encryption methods, but their level of security and privacy varies significantly. While transport layer encryption (TLS) is commonly used across all platforms to secure messages during transit, the implementation of end-to-end encryption (E2EE), which ensures that only the communicating users can access message content, is not universally adopted. For example, WhatsApp and Facebook Messenger provide E2EE by default for all communications, offering a high level of privacy. In contrast, platforms like Telegram only offer E2EE as an optional feature, requiring users to enable it manually for specific chats. Other platforms, such as WeChat, Discord and Teams, do not provide E2EE for messaging exchange, leaving message content accessible to the service provider.

Despite encryption, prior work has demonstrated that encrypted traffic still leaks information through side-channel features such as packet sizes, timings, and flow statistics [3–5]. Thus, while end-to-end encryption secures message content from eavesdroppers, it does not entirely protect user privacy, remaining a critical concern in the use of IM platforms. The activity associated with messaging (e.g. as sending and receiving texts or media) could still be inferred through traffic analysis. This type of analysis (referred to as activity fingerprinting), which uses observable traffic features to infer user actions or application states, can

* Corresponding author.
*E-mail addresses:* lmehavilla@unizar.es (L. Mehavilla), jogarmo@unizar.es (J. García), alesanco@unizar.es (Á. Alesanco).

https://doi.org/10.1016/j.knosys.2026.115893
Received 31 March 2025; Received in revised form 19 March 2026; Accepted 26 March 2026
Available online 30 March 2026

**Table 1**
Instant message platforms studied.

| Platform | Users/ million | Transport Encryption | End-to-End Encryption (E2EE) | E2EE Availability |
|---|---|---|---|---|
| Discord [11] | 563 | TLS | Partial (available for audio and video calls) | Messages are accessible to the provider |
| Facebook Messenger [1] | 1010 | TLS | Default for all communications | Available by default |
| Instagram [12] | 1400 | TLS | Partial (testing E2EE for direct messages) | Not fully available yet |
| Snapchat [1] | 800 | TLS | Partial (snaps are E2EE, but text messages are not) | Not available for all communication types |
| Teams (Microsoft Teams) [13] | 320 | TLS | Partial (available for one-on-one VoIP calls) | Not available for all messaging features |
| Telegram [1] | 900 | MTProto | Optional (via 'Secretes Chats') | Must be manually enabled for individual chats |
| WeChat [1] | 1343 | MMTLS | No E2EE | Messages are accessible to the provider |
| WhatsApp [1] | 2000 | TLS | Default for all communications | Available by default |
| X (formerly Twitter) [14] | 540 | TLS | Partial (testing E2EE for direct messages) | Not fully available yet |

expose sensitive information about a user's behaviour, communication habits, and interaction patterns.

Given the complexity of these patterns, machine learning (ML) techniques are particularly well suited for uncovering and analysing them, enabling more accurate identification of specific activities despite the encryption in place. They have proven to be powerful tools across various domains, particularly in the automatic classification of actions and behaviours based on data patterns present in encrypted network traffic [6]. This effectiveness is due to machine learning's ability to process vast amounts of data and uncover hidden insights, as demonstrated by its high accuracy rates in applications such as intrusion detection in IoT systems [4] and fingerprinting in encrypted network traffic [7–9]. However, as highlighted in [10], achieving satisfactory results with machine learning requires a systematic approach, including data collection, feature extraction, feature reduction and selection, algorithm selection, model construction, and validation.

Existing approaches to IM traffic analysis still present important limitations. Many are tested on a single platform, which limits their ability to generalize across heterogeneous ecosystems. Others depend on long observation windows and offline processing, making them unsuitable for real-time scenarios. In addition, models that achieve high accuracy often do so at the cost of computational complexity, hindering their deployment in latency-sensitive or resource-constrained environments. Together, these factors highlight the need for lightweight and adaptable ML-based solutions capable of operating real time while maintaining cross-platform generalization.

We hypothesize that employing short transaction windows aligned with activity boundaries, selective enrichment with features from concurrent secondary connections, and platform-aware feature selection collectively yield a stable and highly discriminative representation of IM traffic. Using this representation, lightweight ML classifiers, specifically, tree-based models, can accurately map encrypted transactions to user actions with both high precision and low latency. To support this study and future research, we also created a novel dataset of IM traffic spanning nine major platforms and multiple user actions.

Thus, this work pursues three main objectives. First, it aims to determine whether user activities can be accurately characterized from short, transaction-aligned fragments of encrypted IM traffic using lightweight ML models. Each burst of packets associated with a specific user action is treated as a transaction, capturing the temporal and volumetric features most representative of that activity while minimizing background noise. Second, the study explores whether such transaction-aligned representations, selectively enriched with concurrent-connection context and platform-aware features, can enable accurate and low-latency recognition of user actions. To this end, we formalize a segmentation strategy aligned with user activity boundaries and enrich each transaction with features extracted from concurrent secondary connections. Finally, the third objective is to design lightweight ML classifiers suitable for real time operation. For this purpose, we implement and test models optimized for efficiency and adaptability using traffic captured from IM platforms.

The nine IM platforms analysed in this study (Discord, Facebook Messenger, Instagram, Snapchat, Microsoft Teams, Telegram, WeChat, WhatsApp, and X) present diverse connection topologies and background behaviours. Table 1 summarizes their main communication characteristics, highlighting the diversity and representativeness of the dataset used. The selection of platforms covers differences in encryption schemes, ensuring that the proposed method is evaluated under realistic and heterogeneous conditions that reflect the challenges of modern IM ecosystems.

The main contributions of this paper are:

- **Transaction-centric segmentation for real time action recognition.** We introduce a segmentation strategy that aligns short transactions with user activities, enabling real-time classification without the drawbacks of fixed-window aggregation.
- **Secondary-transaction enrichment.** We demonstrate how concurrent connections can provide complementary evidence and selectively integrate them to improve robustness in some platforms.
- **ML-based classification of transaction patterns.** We demonstrate that lightweight tree-based classifiers effectively map encrypted transaction features to user actions, balancing accuracy, and efficiency.
- **Cross-platform evaluation.** We conduct a large-scale study on nine IM platforms, validating the generality and robustness of our pipeline.
- **Dataset creation.** We contribute a new dataset of IM traffic traces, covering multiple platforms and action types.
- **Platform-aware feature selection.** Using information gain, we derive compact feature sets that retain discriminative power while reducing computational overhead.
- **Comprehensive analysis and ethical discussion.** We analyze per-platform results, investigate limitations in challenging cases, and discuss ethical implications and threats to validity for real-world use.

The rest of the paper is organised as follows. In Section 2 a review of the literature on the topic is made. In Section 3 the materials and methods that were used are presented. In Section 4 the results obtained are shown and discussed. Finally, in Section 5 the ethical and privacy concerns are addressed and in Section 6 the conclusions of our research are enumerated.

## 2. Literature review

The identification of user activities from encrypted instant messaging traffic has evolved through several methodological stages, reflecting different assumptions about data accessibility, privacy, and scalability. Early research primarily relied on direct access to application databases stored on user devices, while later studies shifted toward analysing observable network traffic to infer user behaviour without decrypting content. More recent work has incorporated machine learning to enhance recognition accuracy and automation. This section provides a critical and structured review of these approaches.

### 2.1. Database-based analysis of messaging activity

Early attempts to supervise messaging activity relied on examining local application databases stored on smartphones. For instance, [15] and [16] demonstrated that WhatsApp's database contains a wealth of information, including details about messages, recipients, and contact list modifications, among other data. While this approach allows reconstruction of user activity timelines, as shown in [16], it suffers from severe ethical and practical limitations. Participants must physically surrender their devices for inspection, which disrupts natural usage patterns, provides only a temporary snapshot of behaviour, and exposes message content in plaintext. These privacy and consent concerns make database inspection intrusive, non-scalable, and ethically questionable, limiting its feasibility for continuous or remote analysis.

### 2.2. Network-traffic-based approaches

In response to the limitations of database inspection, research attention shifted toward network-traffic analysis, where observable flow features are used to infer user activity without accessing content. This research stream can be categorized into four main methodological lines: (i) early flow characterization, (ii) post-connection and offline analysis, (iii) flow segmentation and event-based methods, and (iv) fine-grained activity fingerprinting frameworks.

#### 2.2.1. Early flow characterization

Initial network-based studies examined general traffic behaviour rather than specific user actions. For instance, Fiadino et al. [17] analysed WhatsApp network connections to extract Quality of Service (QoS) indicators, while [5] studied WeChat's MMTLS traffic to observe how flow characteristics vary with user activity. Although [5] employed machine learning for traffic flow classification, the considered activities (e.g., payments, browsing, subscripting) were not related to typical messaging. These studies confirmed that encrypted flows carry identifiable statistical patterns, but they were restricted in scope and did not explore user-level activity recognition.

#### 2.2.2. Post-connection and offline analysis

Another research line examined traffic only after sessions concluded, limiting real-time applicability. For example, Ramraj and Usha [18] classified normal versus malicious WhatsApp traffic using SSL packet lengths and ML models, while [19] attempted to infer activities from post-connection traffic traces of WhatsApp and Telegram through manual inspection and predefined rules. This offline, rule-based method demonstrated feasibility but lacked scalability, automation, and responsiveness, preventing real-time or large-scale deployment.

#### 2.2.3. Flow segmentation and event-based methods

Later works introduced segmentation to isolate bursts or events within network flows. In [3], packet bursts were grouped into events when inter-packet delays were below a threshold, filtering out protocol noise. While effective for detecting general activity periods, this method was focused on identifying channel members, not on discerning actions occurring on those events.

Similarly, Conti et al. [20] introduced a framework for analysing encrypted network traffic from seven mobile applications (Gmail, Facebook, Twitter, Tumblr, Dropbox, Google+, and Evernote), applying timeout-based segmentation and time-series byte counts to classify user-initiated actions with ML models. Nevertheless, the approach focused on actions initiated by the user, such as sending messages, posting or opening a page, and ignored received messages and notifications, preventing symmetrical modelling of communication. The framework in [21] identified user actions in WeChat and WhatsApp, extending segmentation by introducing hierarchical structures (flow → session → dialog) for WeChat and WhatsApp. Yet it did not handle multiple simultaneous connections and required full connection observation. Moreover, each detected usage combined both the sending and reception of messages.

Likewise, Li et al. [22] proposed a systematic solution for identifying app activity and the actions occurring within Android applications. It used combinatorial optimization to segment flows into bursts, but only after full session completion, precluding real-time identification. Additionally, the study is primarily focused on app recognition, providing limited explanation of how the activity detection results were obtained and demonstrating only a small proof of concept based on user-generated Twitter activities. Collectively, these methods advanced temporal segmentation but remained dependent on complete flow observation, hindering real-time inference, and limited cross-platform comparison.

#### 2.2.4. Activity fingerprinting and multi-application frameworks

More recent research adopted ML pipelines for fine-grained activity identification. PacketPrint [8] combined sequential XGBoost segmentation with hierarchical Bag-of-Words models and binary classifiers to identify app activities from 802.11 frames. Although innovative, this approach was limited to a predefined set of user-generated actions (e.g., browsing, calling) derived from UI components, not encompassing all possible activities.

Similarly, NetScope [23] used short 5 ms capture windows, extracted statistical features from IP headers, and applied a two-step pipeline (K-means + SVM) to classify activities. While capable of detecting broad app usage, NetScope was restricted to coarse-grained categories such as "chat activity" and lacked resolution for distinguishing specific messaging operations. These frameworks illustrate progress in applying ML to encrypted traffic but remain computationally demanding, cover sent actions only, and restrict real-time operation and cross-platform comparison.

### 2.3. Summary and research gap

Table 2 presents a comparative summary of the related work on encrypted traffic analysis for instant messaging activity fingerprinting, alongside our study. It enables comparison of methods, real-time performance, labelling consistency, and main limitations.

Across the literature, several consistent gaps remain unaddressed:

- Dependence on long or complete connection traces, which prevents real-time detection.
- Lack of consistent labelling, arising from the use of differing labels or activity definitions across platforms, thereby limiting cross-platform comparability.
- Heavy, opaque ML models that impede deployment in low-latency or resource-constrained environments.
- Neglect of concurrent connections and background traffic, which characterize modern IM platforms.
- Limited ethical safeguards and reproducibility, with many datasets and parameters unavailable or based on intrusive data collection.

The present work addresses these shortcomings by introducing a unified, real-time framework that identifies user actions (including both sending and receiving messages) from encrypted traffic across nine IM platforms. Our method combines transaction-centric segmentation, selective enrichment of secondary connections, and lightweight ML models, offering a scalable and generalizable approach that overcomes the practical and ethical limitations of prior research.

Unlike most studies, which focus on security-related tasks [6] or rely on heterogeneous activity labels across applications, this study addresses the underexplored problem of routine user-action recognition. Moreover, the framework supports real-time detection, overcoming the common challenge of processing latency in conventional methods [24], and extends the application of machine learning beyond traditional uses such as intrusion or anomaly detection [25,26].

**Table 2**
Comparative summary of related work in encrypted traffic analysis for instant messaging activity fingerprinting. Note: "Real time" refers to the capability of a method to identify network activity as it occurs and "Consistent Labelling" indicates whether the methodology maintains uniform class labels across different platforms or applications.

| Ref. | Platform(s) | Method / Approach | Real-Time | Consistent Labelling | Main Limitation(s) |
|---|---|---|---|---|---|
| [15],[16] | WhatsApp | Local database inspection on smartphones; extraction of user activity timelines | ✗ | Not applicable | Requires physical access to devices; intrusive and non-scalable; exposes plaintext messages and metadata. |
| [17] | WhatsApp | Flow-level QoS analysis (bytes exchanged, durations, patterns). | ✗ | Not applicable | Descriptive analysis only; does not infer specific user actions. |
| [5] | WeChat | Statistical analysis of MMTLS traffic with ML classification of app activities. | ✗ | ✓ | Focused on non-messaging actions (payments, ads) and classified traffic flows. |
| [18] | WhatsApp | SSL packet-length analysis to distinguish normal vs. malicious traffic using ML models. | ✗ | ✓ | Aimed at malicious detection rather than user activity inference; post-connection only. |
| [19] | WhatsApp, Telegram | Manual post-connection inspection; rule-based association of packet patterns with actions. | ✗ | ✓ | Manual, heuristic, and non-automated; requires full session capture; low scalability. |
| [3] | IM apps | Event extraction via packet-burst detection based on inter-packet delay threshold. | Partial | Not applicable | Identifies bursts but not specific actions within them. |
| [20] | Gmail, Facebook, Twitter, Tumblr, Dropbox, Google+, Evernote | Timeout-based segmentation; ML classification of user-initiated actions. | ✗ | ✗ | Considers only sent actions; no received message handling; limited cross-app comparability. |
| [21] | WeChat, WhatsApp | Hierarchical segmentation (flow → session → dialogue) combined with ML classification. | ✗ | ✓ | Requires full flow observation; merges send/receive actions. |
| [22] | Android apps (e.g., Twitter) | Flow division into bursts via combinatorial optimization and decision logic. | ✗ | ✗ | Needs complete flow capture; limited proof of concept; weak methodological explanation. |
| [8] | 802.11 frames | PacketPrint: Sequential XGBoost segmentation + Hierarchical Bag-of-Words + Binary classifiers for activity fingerprinting. | Partial | ✗ | Considers only user-generated actions derive from UI components; computationally heavy. |
| [23] | Multiple smartphone apps (e.g., Snapchat, WhatsApp) | NetScope: short 5 ms windows, 26 statistical features, K-means + SVM two-stage classification. | ✗ | ✓ | Detects broad app usage only; lacks fine-grained action distinction; resource-intensive. |
| **This work** | 9 IM platforms (Discord, Facebook Messenger, Instagram, Snapchat, Microsoft Teams, Telegram, WeChat, WhatsApp, and X) | transaction-centric segmentation, secondary-connection enrichment, and lightweight tree-based ML classifiers | ✓ | ✓ | Residual challenges with highly dynamic background traffic (e.g., Instagram, X); otherwise robust, generalizable and consistent labelling across platforms. |

## 3. Materials and methods

### 3.1. Characterization

This subsection presents an exploratory characterization of encrypted instant messaging traffic to understand the structure, type, and temporal dynamics of the network connections involved. The insights obtained here form the empirical foundation for the segmentation and classification framework. All key parameters and thresholds were derived from measured traffic statistics obtained during the manual characterization stage, ensuring reproducibility, transparency, and cross-platform robustness.

#### 3.1.1. Exploratory analysis and data acquisition

To conduct a preliminary characterization of the traffic generated by these messaging platforms, we captured and visually inspected 50 real connections for each platform, each lasting 10 s and encompassing various types of message exchanges. All captures were performed in controlled environments without intercepting third-party communications, ensuring full ethical compliance.

Recognizing that messaging platforms employ different transmission mechanisms depending on the device or environment, we established connections using a variety of software, including mobile apps, web interfaces accessed via different browsers, and desktop applications running on multiple operating systems. For each connection, we derived a packet timeline, with each packet represented by a column whose height was proportional to its size (see Figs. 1 and 2 for examples from WhatsApp and X, respectively).

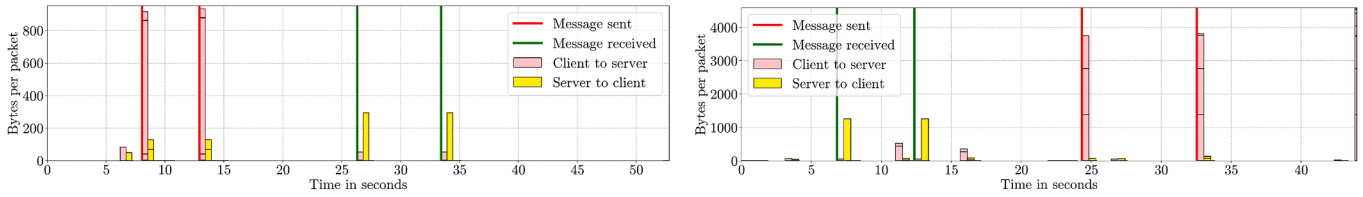
#### 3.1.2. Connection taxonomy and behavioural patterns

After careful inspection, several key conclusions were reached. As anticipated, no significant variations in behaviour were observed when sending or receiving messages across different software environments. The connections remained consistent regardless of the underlying operating system, mobile app, web environment, or desktop version used. Additionally, we identified three distinct types of connections that appeared consistently across all platforms.

**Principal Connection:** This connection initiates as soon as a user accesses the platform and persists until the user exits. It includes signalling data associated with the transmission of messages, both sent and received. Typically, it is characterised by a low data rate and long duration.

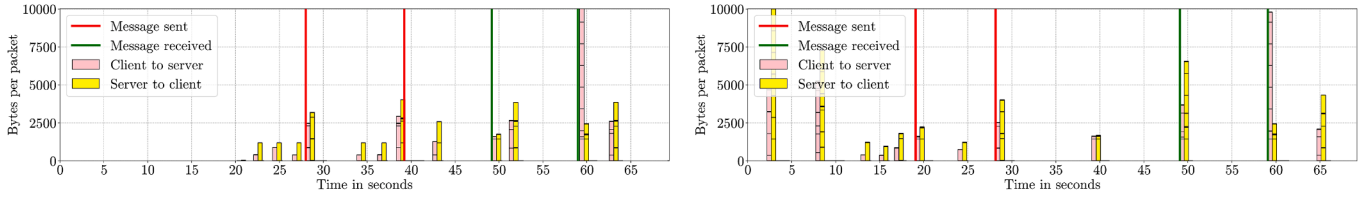
**Secondary Connection:** Triggered exclusively when a file is being uploaded to or downloaded from the server, such as audio files, media content, or documents. It is generally marked by heavy data flows and short duration.

**Control Connection:** Responsible for facilitating the exchange of statistics and application-specific information, including the transmission of short-lived, temporary private keys. These connections are short-lived and involve the exchange of small packets.




(a) Text messages. (b) Multimedia messages.

**Fig. 1.** Example of message signalling in a WhatsApp principal connection (Plotting packets with more than 100 bytes).




(a) Text messages. (b) Multimedia messages.

**Fig. 2.** Example of message signalling in an X principal connection (Plotting packets with more than 100 bytes).

It was particularly intriguing to observe that message transmission patterns could often be visually inferred along principal connections, though this was not equally straightforward across all platforms. WhatsApp exhibited the most discernible and structured patterns, making it the most suitable for visual analysis, whereas X proved to be the most challenging platform for pattern identification. Figs. 1 and 2 illustrate these visual patterns for different message types on WhatsApp and X, respectively. The remaining platforms fell between these two extremes.

### 3.1.3. Temporal behaviour and transaction segmentation

It was observed that within the packet timeline (traffic flow) that periods of activity and inactivity were alternated. All connections could therefore be divided into discrete transactions based on the time interval between consecutive packets. A transaction is defined as a period during which a single message is either transmitted or received, or a period of inactivity during which no information is exchanged. Consequently, when the time interval between two packets exceeds a certain threshold, it marks the end of one transaction and the start of a new one. This segmentation approach is particularly advantageous because it enables real-time processing of encrypted traffic.

Determining the optimal threshold for segmentation was crucial. Setting the threshold too high would result in multiple independent messages being grouped into a single transaction [20], reducing accuracy in message type discrimination and message count estimation. Conversely, setting it too low could cause packets belonging to the same message to be split across separate transactions [3], complicating correct message identification.

To ensure that the selected time threshold was not arbitrary, we analysed the temporal statistics obtained from the 50 inspected connections (see Section 3.1.1). Specifically, we calculated two metrics: (i) the maximum inter-packet interval observed within transactions ($\max(\Delta t_{\text{intra}})$), and (ii) the minimum inter-packet interval between consecutive transactions ($\min(\Delta t_{\text{inter}})$). These two quantities define a separation window in which packets belonging to the same message remain temporally correlated, while packets beyond this range indicate the start of a new transaction. Additionally, the mean intra-transaction and inter-transaction intervals ($\overline{\Delta t_{\text{intra}}}$ and $\overline{\Delta t_{\text{inter}}}$) were computed to evaluate the overall compactness and periodicity of message transmissions.

Across the inspected samples, the values of $\max(\Delta t_{\text{intra}})$ and $\min(\Delta t_{\text{inter}})$ were found to be 0.97 s and 1.04 s, respectively, resulting in an extremely narrow separation window of only 0.07 s. The mean intra-transaction interval ($\overline{\Delta t_{\text{intra}}} = 0.0166$ s) and mean inter-transaction interval ($\overline{\Delta t_{\text{inter}}} = 5.8441$ s) further confirm the strong temporal contrast between message bursts and distinct transactions. Although the boundary between the maximum intra- and minimum inter-intervals is tight, the large difference in their mean values indicates that intra-transaction activity remains highly compact and well separated in time from subsequent transactions. Consequently, a threshold of 1 s was adopted as a robust and representative value, balancing segmentation precision with temporal consistency across platforms. This result reinforces the statistical separability of intra- and inter-transaction timings, supporting the validity of a global time threshold for real-time message segmentation.

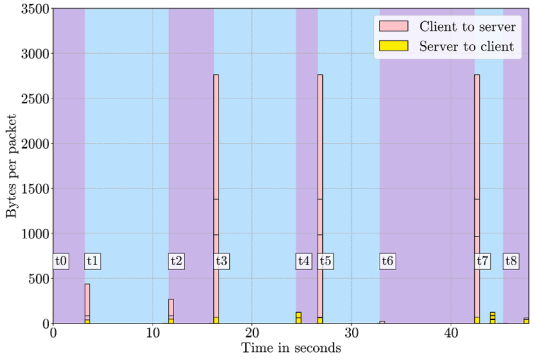



**Fig. 3.** Example of a WhatsApp principal connection segmented into transactions. Each coloured stripe represents a transaction labelled with identifier t* (from t0 to t8).

Fig. 3 presents an example of a WhatsApp principal connection segmented into eight transactions according to this criterion. The connection was captured in a real-world scenario involving three transmitted messages. Each coloured stripe represents a transaction labelled with the identifier t*. The figure shows that inter-packet intervals within a transaction are consistently less than one second, while transaction boundaries occur when the interval exceeds this threshold.

The analysis further revealed two distinct types of transactions: those corresponding to periods of inactivity (t0, t1, t2, t4, t6, and t8) and those linked to user actions (t3, t5, and t7). Interestingly, during active user conversations, the number of inactivity transactions (hereafter referred to as StandBy) often exceeded those representing user actions. The magnitude of this imbalance varied between platforms, an observation that later guided the design of our datasets to replicate this behaviour accurately.

#### 3.1.4. Summary of observations and design implications

The characterization presented in this subsection established the foundation for our subsequent segmentation and classification framework. By identifying the structural and temporal properties of encrypted IM traffic, we confirmed that such traffic can be effectively represented as a series of short, well-defined transactions distributed across multiple concurrent connections. These findings directly motivated the empirical classification and feature-extraction strategy detailed in Section 3.2, ensuring that the design of our framework is data-driven and grounded in measurable traffic behaviour.

### 3.2. Feature extraction and transaction classification

This subsection describes the extraction of transaction-level features used as input to the machine learning algorithm responsible for identifying user actions within each transaction. The process was implemented using the Zeek scripting language and involved the segmentation of connections into transactions, extraction of statistical features, and classification of connection types. These procedures collectively form the basis of our data-driven representation of encrypted instant messaging traffic.

#### 3.2.1. Transaction-level data collection with zeek

Zeek is a powerful network security analysis framework capable of monitoring network traffic and generating detailed connection logs [27]. By default, it produces the *conn.log* file, which records detailed metadata for all concluded connections, including IP addresses and ports, transport protocol, connection state and history, duration, and bytes or packets transmitted and received.

However, analysing entire connections is insufficient for our purpose, as complete connection-level data does not allow for distinguishing individual user messages. To address this limitation, we developed a custom Zeek script that segments connections into the individual transactions defined in Section 3.1. This segmentation allows for fine-grained representation of traffic bursts corresponding to discrete user actions.

#### 3.2.2. Identification of platform connections

To accurately associate network connections with their corresponding IM platforms, a reference database was constructed linking known server IP addresses to each service. This database serves as a dynamic mapping resource that enables consistent identification of platform-related traffic even as infrastructures evolve. Given that messaging providers frequently modify or expand their server networks, due to content delivery optimizations, regional deployments, or version updates, the database is designed to be regularly maintained and updated to incorporate newly observed IP addresses and to remove deprecated ones. This continuous update process ensures that platform attribution remains accurate and reliable over time. During capture, our Zeek script continuously inspects incoming packets, filtering those belonging to these identified IPs and tracking relevant statistics in real time. The script also determines when a transaction ends and a new one begins, using the one-second inter-packet threshold defined in the previous section. Once a transaction concludes, Zeek automatically extracts and stores its statistical features.

#### 3.2.3. Feature definition and classification of connection types

For each transaction, we recorded the duration, total bytes and packets sent or received, and other attributes derived from Zeek's connection metadata. Some information typically found in *conn.log*, such as IP addresses, port numbers, or connection state, was excluded, as these are not discriminative for transaction-level analysis and, in many cases, incomplete (since connections may not be terminated).

To enhance model performance, we extended Zeek's standard fields by computing additional statistical descriptors within our custom script. Table 3 lists the complete set of parameters extracted per transaction. These features capture both the intensity and variability of packet exchanges within a transaction, providing a compact yet expressive numerical representation suitable for machine learning classification.

The extraction process was applied across all transactions from principal, secondary, and control connections. Transactions are classified as new when they are the first identified within a connection or established when related to an already active connection. To associate each new transaction with a connection type, we employed a straightforward yet robust rule grounded in the behavioural analysis of Section 3.1.

We continuously track active connections and allow only one principal connection per communication session with a messaging platform. If no principal connection is active, the next new transaction is classified as belonging to the principal connection and marked as active. If a principal connection is active, all subsequent new transactions are treated as secondary connections unless the transaction's total byte sum is below the empirically derived threshold of 20 kB, in which case it is classified as a control connection.

To calculate the threshold, we analysed the byte statistics obtained from the 50 inspected connections. Specifically, we calculated two reference metrics: (i) the maximum total bytes observed within control connections ($\max(B_{\text{control}})$), and (ii) the minimum total bytes observed within secondary connections ($\min(B_{\text{secondary}})$). These quantities define a separation window that reflects the natural discontinuity in data volume between both connection types. Across the inspected samples, $\max(B_{\text{control}})$ and $\min(B_{\text{secondary}})$ were found to be 14.9 kB and 25.1 kB, respectively, resulting in a non-overlapping gap of approximately 10 kB. The threshold was then calculated as the midpoint between these two empirical boundaries:

$$B_{\text{threshold}} = \frac{\max(B_{\text{control}}) + \min(B_{\text{secondary}})}{2} \approx 20 \text{ kB}$$

This approach yields a representative and robust value of 20 kB, ensuring consistent separation between control and secondary connections. The mean values within each group further support this differentiation: control connections average less than 5.3 kB per transaction, whereas secondary connections exhibit average data exchanges exceeding 10.6 MB. Consequently, the selected threshold reflects an intrinsic statistical property of the traffic rather than an arbitrary configuration. As with the time-based segmentation criterion, this empirical procedure can be repeated whenever platform behaviour evolves, ensuring adaptability and reproducibility across future studies.

Thus, secondary transactions can provide complementary evidence when occurring concurrently with principal ones, particularly during multimedia exchanges. For example, when an image is transmitted, signalling data travel over the principal connection while the media file itself is transferred through a secondary connection. To capture this interaction, we enrich the principal transaction with information from any secondary transaction occurring within 2.5 s of it. No secondary connection was observed beyond this value. This enrichment rule was derived empirically from the temporal characterisation described in Section 3.1.

#### 3.2.4. Robustness and generalizability of the approach

To support reproducibility, a reference database linking known server IP addresses to their respective IM platforms was constructed and is periodically updated, ensuring accurate identification of platform endpoints despite ongoing infrastructure or deployment changes. However, the distinction between connection types does not rely on IP-based identification. Instead, the classification process is governed exclusively by a set of behavioural and statistical rules derived from message dynamics and traffic patterns. This rule-based design guarantees that the method remains robust even when IP addresses or port numbers are reassigned. For instance, when a server previously used for a principal connection later hosts a secondary one. Consequently, the approach is both interpretable and platform-agnostic, offering long-term reliability and adaptability across diverse and evolving IM environments, while addressing one of the key limitations identified in prior literature.

**Table 3**
Transaction parameters extracted per transaction.

| ID | Name | Description |
|---|---|---|
| 1 | meanByteSent | Mean payload bytes per packet sent (excluding packets with a payload of 0). |
| 2 | meanByteRx | Mean payload bytes per packet received (excluding packets with a payload of 0). |
| 3 | stdByteSent | Standard deviation of payload bytes per packet sent (excluding packets with a payload of 0). |
| 4 | stdByteRx | Standard deviation of payload bytes per packet received (excluding packets with a payload of 0). |
| 5 | pktSent | Total number of packets sent with non-zero payload. |
| 6 | pktRx | Total number of packets received with non-zero payload. |
| 7 | meanTime | Mean interval time between packets. |
| 8 | stdTime | Standard deviation of interval time between packets. |
| 9 | time | Duration (in seconds) during which the packets were transmitted. |

### 3.3. Data collection

Constructing a dataset for training and testing machine learning algorithms is essential to determining the most effective approach for instant messaging detection. To achieve this, we meticulously created dedicated datasets for each platform studied, as outlined in Table 4. The data generation process was comprehensively designed to replicate typical user behaviour, including typing messages and exchanging common multimedia content like audio and images.

#### 3.3.1. Traffic generation and capture setup

The data generation process began with two Android Studio phone emulators accessing the respective platforms, while a dedicated Python script controlled their keyboard and touch inputs. The script was carefully developed to emulate realistic user actions, including varying typing speeds and the specific steps required to send multimedia messages in real time. This approach allows the generation of labelled traffic from real instant messaging platforms, reflecting authentic network patterns without requiring real users. Although the user actions are simulated, the resulting traffic captures real-world network behaviours and interactions, providing a realistic and representative dataset. To further demonstrate the realism of the generated dataset, the best-performing WhatsApp model was evaluated using traffic captured from an actual conversation between two users exchanging 30 text messages.

#### 3.3.2. Transaction capture and feature enrichment

The encrypted network traffic from both emulated phones was captured and analysed using Zeek. The features of each transaction were categorised based on the connection they belonged to (as explained in Section 3.2.3). If a transaction was identified as part of the principal connection, it was further enriched with information from a secondary connection transaction, provided the secondary transaction occurred within 2.5 s of the principal transaction, as previously explained. Thus, the feature vector contains both arrays of features if enriched, or only the principal transaction features expanded with zeros.

#### 3.3.3. Automatic labelling and dataset composition

These transactions were compiled into a dataset and automatically labelled according to predefined classes, as shown in Table 4. The Python script facilitated this labelling process by tracking the timestamp of each sent message. The SB label (StandBy) was assigned to transactions with no activity, while other labels indicated whether a message was received (ending in R) or sent (ending in S). Messages were categorised as either text messages (T) or multimedia messages (MI), with multimedia messages typically generating a secondary connection.

This procedure allowed us to create realistic datasets, as the platforms generated traffic flows representative of typical user behaviour during message exchanges. As noted in Section 3.1, StandBy transactions (SB) are more prevalent than other types, although their frequency varies depending on the platform.

**Table 4**
Datasets instances for each tested platform.

| Platform | Label | Instances | Platform | Label | Instances |
|---|---|---|---|---|---|
| Discord | MIR | 881 | Facebook Messenger | MIR | 982 |
| | MIS | 999 | | MIS | 1008 |
| | SB | 3930 | | SB | 3421 |
| | TR | 678 | | TR | 967 |
| | TS | 1033 | | TS | 1005 |
| Instagram | MIR | 980 | Snapchat | MIR | 1095 |
| | MIS | 1047 | | MIS | 1110 |
| | SB | 3925 | | SB | 8355 |
| | TR | 1044 | | TR | 1005 |
| | TS | 926 | | TS | 1005 |
| Teams | MIR | 992 | Telegram | MIR | 1104 |
| | MIS | 948 | | MIS | 1112 |
| | SB | 4724 | | SB | 14910 |
| | TR | 1003 | | TR | 1003 |
| | TS | 1005 | | TS | 1031 |
| WeChat | MIR | 879 | WhatsApp | MIR | 1075 |
| | MIS | 875 | | MIS | 1111 |
| | SB | 2030 | | SB | 8889 |
| | TR | 916 | | TR | 1072 |
| | TS | 878 | | TS | 1047 |
| X | MIR | 1251 | MIR | Multimedia Reception | |
| | MIS | 1245 | MIS | Multimedia Sending | |
| | SB | 7670 | SB | StandBy | |
| | TR | 929 | TR | Text Reception | |
| | TS | 593 | TS | Text Sending | |

### 3.4. Feature selection

#### 3.4.1. Information gain and feature ranking methodology

In the context of classifying datasets, entropy measures the degree of information among the classes within the data. Information gain, on the other hand, quantifies how much a specific feature reduces this entropy when the data is split based on that feature. A higher information gain suggests that the feature is more valuable for decision-making in a classification model. Mathematically, information gain is calculated as follows:

$$IG(Class, Feature) = H(Class) - H(Class|Feature)$$

Where:

$H(Class)$ represents the entropy of the dataset with respect to the class variable.

$H(Class|Feature)$ represents the conditional entropy of the dataset after it has been split based on the value of the feature, essentially measuring the remaining disorder.

To understand the factors influencing multi-class classification within each platform's dataset, we conducted a feature ranking analysis based on information gain, using the datasets outlined in Table 4. This analysis allowed us to identify which features have the most significant impact on classification outcomes and to detect any discrepancies among the platforms. The feature rankings, determined by the information gain for each feature across platforms, are presented in Table 5. A feature name ending in _conn2 belongs to an associated secondary transaction. Otherwise, it is associated with a principal transaction.

#### 3.4.2. Platform-specific feature relevance and analysis

As shown in Table 5a and c, in Discord and Instagram, the features from secondary transactions are not relevant, with information gains close to zero, whereas features from principal transactions exhibit information gains exceeding 0.3. Similarly, in Facebook Messenger and WhatsApp (see Table 5b and h, respectively), there is a modest information gain gap of 0.05 between features from principal and secondary connections. Conversely, Snapchat and Teams (see Table 5d and e, respectively) display a larger gap of 0.13 points, indicating a greater discrepancy between the significance of these features.

In contrast, secondary connection features in Telegram (see Table 5f) are notably relevant, with most features showing information gains above 0.15, indicating their significant role alongside principal transaction features in the rankings. A similar pattern is observed in X (see Table 5i), where secondary transaction features are intermingled with those from principal transactions, though their information gains are lower than in Telegram, which explains poorer classification performance. Moreover, in WeChat (see Table 5g), all features, including those from secondary transactions, are highly relevant, with even higher information gains, indicating their substantial usefulness in the classification task.

In conclusion, the results demonstrate that in Telegram, X, and WeChat, considering features from associated secondary transactions for training the machine learning algorithms is crucial. In contrast, for the other platforms, secondary transaction features may not be as relevant to distinguish different classes. Therefore, for Telegram, X, and WeChat, our machine learning models will incorporate both principal and secondary associated transactions, while for the other platforms, models will be built using only principal transaction features.

### 3.5. Algorithm selection and model construction

Each machine learning algorithm offers its unique strengths, and the selection of the most suitable algorithm depends on the specific nature of the data and the balance between predictive accuracy and computational complexity. To determine the optimal algorithm for the message detection task, a detailed evaluation was conducted, considering the metrics outlined in Table 6.

Prior to training and testing all machine learning algorithms, an experiment was performed to assess the appropriate amount of data needed to train and test a model effectively. This experiment involved training a Random Forest model with 100 decision trees, varying the proportion of training data between 30% and 80% (in 10% increments) of the dataset, with the remaining data used for testing. The results suggested that a 50/50 split between training and testing data provided the most efficient balance, only the 50% and 80% options are reported in the results section for clarity.

Subsequently, various algorithms were trained and tested (Support Vector Machine, Stochastic Gradient Descent, Nearest Neighbour, Nearest Centroid, Gaussian Naïve Bayes, Bernoulli Naïve Bayes, Decision Tree, Boosting with Decision Trees, Bagging with Decision Trees, Random Forest and Multilayer Perceptron). Therefore, for each platform, the three algorithms that present the best results were further trained and optimised by adjusting hyperparameters such as tree depth, the number of trees, and the number of nodes. This optimisation process ensured that each model was finely tuned to achieve the highest possible performance. By carefully calibrating these parameters, the models were able to effectively capture complex patterns in the data while minimising overfitting.

The experiments in this section were implemented using the scikit-learn package in Python, and the code was developed and executed in a Google Colab Python 3 environment.

## 4. Results and discussion

The following subsections present our results in four parts: (i) training size optimization and model selection, (ii) cross-platform performance evaluation, (iii) feature relevance and real-time capabilities, and (iv) comparative analysis with existing works.

### 4.1. Training size optimization and model selection

Table 7 shows the results for training size optimisation. It presents the performance metrics for the Random Forest model, configured with 100 Decision Trees and trained using 50% and 80% of the dataset. As observed, the variation in results across different platforms when adjusting the amount of training data is minimal, with a deviation of less than 1%. However, a notable difference is observed in the test time per transaction, measured in milliseconds, which increases as the amount of training data grows.

The results in Table 7 demonstrate that training the algorithms with 50% of the total dataset is optimal. This balance provides sufficient information for the models to learn patterns while ensuring the training data represents all possible cases. Training with a larger portion, such as 80%, can lead to overfitting, as evidenced by the lower F1-score, recall, and precision metrics. Additionally, the time required to predict a new instance, increases with more training data due to the complexity of the model, which takes longer to make decisions. Therefore, a 50%-50% training-test split ensures that the models generalise well to new data, optimising performance without overfitting and providing reliable predictions across various scenarios.

### 4.2. Performance evaluation across platforms

The performance metrics for the MLP model and the top three techniques, Random Forest, Bagging, and Boosting with Decision Trees, optimised using the Random Search method and trained on 50% of the dataset, with the remaining portion reserved for testing, are shown in Table 8, with the best model highlighted in bold. All platforms demonstrate satisfactory results, achieving over 79% in their performance metrics, with the exceptions of Instagram and X, which do not exceed 65%. In this Table 8 the performance results obtained for an optimised MLP classifier are also included. Lastly, Table 9 displays the confusion matrices for the best-performing model on each platform.

As illustrated in Table 8, it is clear that Decision Tree-based algorithms (Bagging, Boosting, and Random Forest) consistently deliver the best results for detecting user activity on instant messaging platforms. While other algorithms also perform well, their training and testing times are considerably longer. These three models, except those associated with Instagram and X, achieve promising results, with F1-scores exceeding 79% and accuracy rates above 82%. Notably, the number of trees/estimators and their depth vary significantly depending on the platform, suggesting that a single model would not be effective across all platforms.

Furthermore, as shown in Table 9, the most common errors occur when the model misinterprets periods of inactivity as periods of activity and vice versa, though the latter is less frequent. This indicates that while the models may sometimes fail to detect a message exchange, they

**Table 5**
Feature ranking generated by the information gain.

| (a) Discord | |
|---|---|
| Feature | IG |
| meanByteSent | 0.8289 |
| meanByteRx | 0.7689 |
| stdByteSent | 0.7002 |
| pktSent | 0.6099 |
| time | 0.5937 |
| stdByteRx | 0.5530 |
| pktRx | 0.5504 |
| meanTime | 0.4067 |
| desvTime | 0.3526 |
| meanTime_conn2 | 0.0372 |
| time_conn2 | 0.0291 |
| pktRx_conn2 | 0.0265 |
| meanByteSent_conn2 | 0.0254 |
| pktSent_conn2 | 0.0244 |
| meanByteRx_conn2 | 0.0243 |
| desvTime_conn2 | 0.0163 |
| stdByteSent_conn2 | 0.0083 |
| stdByteRx_conn2 | 0.0050 |
| (b) Facebook Messenger | |
| Feature | IG |
| meanByteSent | 0.8386 |
| meanByteRx | 0.8303 |
| stdByteRx | 0.7402 |
| stdByteSent | 0.5639 |
| time | 0.5252 |
| pktSent | 0.4476 |
| meanByteRx_conn2 | 0.4383 |
| pktRx | 0.4272 |
| meanByteSent_conn2 | 0.4035 |
| desvTime | 0.3949 |
| meanTime | 0.3821 |
| pktSent_conn2 | 0.3301 |
| time_conn2 | 0.2976 |
| meanTime_conn2 | 0.2733 |
| stdByteRx_conn2 | 0.2553 |
| stdByteSent_conn2 | 0.2403 |
| pktRx_conn2 | 0.2401 |
| desvTime_conn2 | 0.2208 |
| (c) Instagram | |
| Feature | IG |
| meanByteRx | 0.6223 |
| meanByteSent | 0.5766 |
| meanTime | 0.5076 |
| stdByteRx | 0.4317 |
| desvTime | 0.4193 |
| time | 0.3935 |
| stdByteSent | 0.3366 |
| pktSent | 0.3185 |
| pktRx | 0.3177 |
| time_conn2 | 0.0992 |
| desvTime_conn2 | 0.0909 |
| meanTime_conn2 | 0.0822 |
| meanByteSent_conn2 | 0.0819 |
| meanByteRx_conn2 | 0.0750 |
| pktSent_conn2 | 0.0748 |
| pktRx_conn2 | 0.07141 |
| stdByteSent_conn2 | 0.0482 |
| stdByteRx_conn2 | 0.0362 |
| (d) Snapchat | |
| Feature | IG |
| meanByteSent | 0.6868 |
| meanByteRx | 0.5987 |
| stdByteRx | 0.5889 |
| stdByteSent | 0.4500 |
| desvTime | 0.3653 |
| meanTime | 0.2848 |
| pktRx | 0.2743 |
| time | 0.1842 |
| pktSent | 0.1581 |
| meanByteSent_conn2 | 0.0296 |

**Table 5**
Continued.

| Feature | IG |
|---|---|
| desvTime_conn2 | 0.0268 |
| meanTime_conn2 | 0.0240 |
| pktSent_conn2 | 0.0200 |
| stdByteSent_conn2 | 0.0175 |
| stdByteRx_conn2 | 0.0130 |
| pktRx_conn2 | 0.0111 |
| time_conn2 | 0.0105 |
| meanByteRx_conn2 | 0.0102 |
| (e) Teams | |
| Feature | IG |
| meanByteSent | 0.6883 |
| meanByteRx | 0.6014 |
| stdByteRx | 0.5914 |
| stdByteSent | 0.4460 |
| desvTime | 0.3653 |
| meanTime | 0.2848 |
| pktRx | 0.2674 |
| time | 0.1843 |
| pktSent | 0.1518 |
| meanByteRx_conn2 | 0.0237 |
| desvTime_conn2 | 0.0225 |
| pktSent_conn2 | 0.02141 |
| stdByteSent_conn2 | 0.0211 |
| meanByteSent_conn2 | 0.0203 |
| pktRx_conn2 | 0.0191 |
| meanTime_conn2 | 0.0165 |
| stdByteRx_conn2 | 0.0114 |
| time_conn2 | 0.0073 |
| (f) Telegram | |
| Feature | IG |
| meanByteRx | 0.4889 |
| stdByteRx | 0.4228 |
| pktRx | 0.2844 |
| pktSent | 0.2770 |
| time | 0.2709 |
| meanTime_conn2 | 0.2425 |
| pktSent_conn2 | 0.2312 |
| meanByteRx_conn2 | 0.2301 |
| desvTime_conn2 | 0.2275 |
| meanByteSent_conn2 | 0.2264 |
| meanByteSent | 0.2167 |
| time_conn2 | 0.2165 |
| meanTime | 0.2044 |
| stdByteSent | 0.1935 |
| pktRx_conn2 | 0.1777 |
| stdByteSent_conn2 | 0.1586 |
| desvTime | 0.1502 |
| stdByteRx_conn2 | 0.0201 |
| (g) WeChat | |
| Feature | IG |
| meanByteSent | 0.8396 |
| meanByteRx | 0.7859 |
| stdByteSent | 0.7068 |
| time | 0.6564 |
| meanByteSent_conn2 | 0.6238 |
| stdByteRx | 0.5815 |
| pktSent | 0.5593 |
| pktRx | 0.5490 |
| meanTime | 0.5318 |
| meanByteRx_conn2 | 0.5190 |
| desvTime | 0.4889 |
| stdByteSent_conn2 | 0.4680 |
| pktSent_conn2 | 0.4469 |
| time_conn2 | 0.4226 |
| stdByteRx_conn2 | 0.3280 |
| pktRx_conn2 | 0.3177 |
| meanTime_conn2 | 0.2976 |
| desvTime_conn2 | 0.2615 |
| (h) WhatsApp | |
| Feature | IG |
| meanByteRx | 0.8913 |

**Table 5**
Continued.

| Feature | IG |
|---|---|
| stdByteSent | 0.8433 |
| meanByteSent | 0.8352 |
| stdByteRx | 0.5248 |
| pktSent | 0.4159 |
| time | 0.4011 |
| pktRx | 0.3892 |
| desvTime | 0.2756 |
| meanTime | 0.2580 |
| meanByteRx_conn2 | 0.1941 |
| pktRx_conn2 | 0.1852 |
| meanByteSent_conn2 | 0.1768 |
| pktSent_conn2 | 0.1738 |
| desvTime_conn2 | 0.1484 |
| time_conn2 | 0.1394 |
| meanTime_conn2 | 0.1324 |
| stdByteRx_conn2 | 0.1043 |
| stdByteSent_conn2 | 0.1028 |
| (i) X | |
| Feature | IG |
| meanByteRx | 0.3333 |
| meanByteSent | 0.2542 |
| pktSent | 0.2011 |
| desvTime | 0.1920 |
| meanByteRx_conn2 | 0.1809 |
| time | 0.1763 |
| meanTime | 0.1726 |
| time_conn2 | 0.1682 |
| stdByteRx | 0.1667 |
| meanByteSent_conn2 | 0.1390 |
| stdByteSent | 0.1254 |
| pktSent_conn2 | 0.1225 |
| pktRx_conn2 | 0.1166 |
| pktRx | 0.1161 |
| meanTime_conn2 | 0.1076 |
| stdByteSent_conn2 | 0.1035 |
| desvTime_conn2 | 0.0876 |
| stdByteRx_conn2 | 0.0752 |

**Table 6**
Performance metrics.

| Metric | Description |
|---|---|
| Recall | Recall measures a model's ability to correctly identify all relevant instances within a dataset. |
| Precision | Precision evaluates a model's accuracy in classifying positive instances among all instances it labels as positive. |
| Accuracy | Accuracy measures the proportion of correct predictions among the total number of cases examined. |
| Error Rate | The error rate is the complement of accuracy and quantifies the proportion of incorrect predictions. |
| F1 Score | The F1 score is a harmonic mean of precision and recall. |
| Inference Time | Inference time quantifies the time taken by a trained model to make predictions on new, unseen data. |
| Confusion matrix | A confusion matrix is a table used to evaluate the performance of a classification algorithm. It shows the real versus predicted classifications. |

correctly identify the type of message when they do. This pattern is consistent across platforms, except for Instagram and X, which produce the worst results. On Instagram (see Figure 9c), periods of inactivity are often mistaken for activity, and the model struggles to identify the message type. Conversely, while X correctly identifies message types, it tends to predict inactivity even when messages are exchanged (see Figure 9i). This discrepancy may be attributed to background processes in these platforms, which increase signalling traffic and obscure user-initiated actions. Instagram and X offer a broader range of in-app activities beyond message exchange, and some of them can generate second-plane traffic even when not actively performed by the user. Such second-plane activities introduce noise and make it more difficult for the model to accurately discern user behaviour.

Deep learning algorithms are often herald for their ability to handle complex data. However, this analysis demonstrates that tree-based methods, particularly Random Forest and Bagging, can outperform deep learning models like MLP classifier in this scenario. This is because tree-based methods are better suited to the nature of the data and the scale of this particular problem. When dealing with a dataset which is not massive, as it is the case, where the patterns are not particularly hidden and the task involves structured data with clear feature relationships, deep learning models might be overkill or less effective. The dataset may not provide enough complexity or the volume necessary to justify their use. Conversely, tree-based models provide a balance accuracy, interpretability, robustness, and computational efficiency, making them a practical and effective solution for many classification tasks. Nonetheless, continued refinement and platform-specific adaptations could further enhance performance, particularly on more challenging platforms like Instagram and X.

### 4.3. Feature relevance and real-time detection capabilities

The feature ranking analysis based on information gain (see Table 5) further clarifies the relationship between machine learning performance

**Table 7**
Performance metrics obtained by Random Forest with 100 Decision Trees, varying the training size between the 50% and 80% of the dataset (the remaining data is used to test).

| Platform | Train size | Prec | Recall | F1 | Acc | Error | Inference time (ms) |
|---|---|---|---|---|---|---|---|
| Discord | 0.8 | 0.90 | 0.89 | 0.89 | 0.90 | 0.09 | 0.27 |
| | 0.5 | 0.91 | 0.89 | 0.90 | 0.91 | 0.08 | 0.15 |
| Facebook | 0.8 | 0.91 | 0.87 | 0.89 | 0.89 | 0.10 | 0.35 |
| Messenger | 0.5 | 0.92 | 0.87 | 0.89 | 0.90 | 0.09 | 0.28 |
| Instagram | 0.8 | 0.66 | 0.62 | 0.64 | 0.69 | 0.30 | 0.21 |
| | 0.5 | 0.66 | 0.63 | 0.64 | 0.70 | 0.29 | 0.15 |
| Snapchat | 0.8 | 0.87 | 0.81 | 0.84 | 0.89 | 0.10 | 0.16 |
| | 0.5 | 0.86 | 0.80 | 0.83 | 0.88 | 0.11 | 0.12 |
| Teams | 0.8 | 0.82 | 0.76 | 0.79 | 0.81 | 0.18 | 0.19 |
| | 0.5 | 0.83 | 0.77 | 0.80 | 0.82 | 0.17 | 0.13 |
| Telegram | 0.8 | 0.90 | 0.78 | 0.83 | 0.93 | 0.06 | 0.18 |
| | 0.5 | 0.90 | 0.79 | 0.84 | 0.93 | 0.06 | 0.14 |
| WeChat | 0.8 | 0.92 | 0.90 | 0.91 | 0.92 | 0.07 | 0.15 |
| | 0.5 | 0.91 | 0.89 | 0.90 | 0.91 | 0.08 | 0.12 |
| WhatsApp | 0.8 | 0.99 | 0.98 | 0.98 | 0.99 | 0.00 | 0.12 |
| | 0.5 | 0.98 | 0.98 | 0.98 | 0.98 | 0.01 | 0.06 |
| X | 0.8 | 0.63 | 0.59 | 0.61 | 0.74 | 0.25 | 0.25 |
| | 0.5 | 0.65 | 0.61 | 0.62 | 0.75 | 0.24 | 0.18 |

metrics and the features extracted from each platform. For instance, WhatsApp, which exhibits superior performance, has features with significant information gain that effectively distinguish user activities. In contrast, X, which shows poorer results, presents features with low information gain. This disparity suggests that the effectiveness of machine learning models is closely tied to the discriminative power of the features on each platform. Understanding these differences can guide future efforts in optimising feature extraction and selection strategies tailored to specific platforms.

We generated and collected data by simulating the behaviour of users exchanging messages on the studied platforms. While this process accurately mimics user behaviour, it is important to acknowledge that it doesn't involve real individuals, which may result in slight variations compared to actual user patterns. In addition, to validate that the generated dataset contains real and representative traffic, we evaluated the WhatsApp Random Forest model described in Table 7 using traffic captured from an actual conversation between two real users who exchanged 30 text messages. This evaluation achieved a F1-score of 0.95, only 0.03 points lower than the 0.98 obtained on the test portion of the generated dataset, demonstrating that the data generation process effectively replicates the users' behaviours.

Lastly, it is interesting to remark that the traffic segmentation approach based on time threshold criteria enables real-time classification. This method allows for classification while packets are being transmitted, rather than waiting for the connections to end, thereby facilitating real-time applications.

### 4.4. Comparative analysis with related work

To assess the competitiveness and realism of the proposed method, we compared its performance against representative studies addressing user activity detection in encrypted traffic (see Section 2). Table 10 summarizes the most directly comparable results, showing that our approach achieves competitive or superior performance across several platforms, despite operating under more diverse and realistic experimental conditions.

In [20], message-sending events on Facebook and Twitter were detected with perfect metrics (1.0 precision, recall, and F1-scores). However, these results were achieved using labelled packet datasets that excluded actions not initiated by the observed user (e.g., message reception), and grouped unrecognised activities into a generic "other" category. This constrained setup simplifies the classification problem but limits realism and generalizability. In contrast, our framework captures a broader range of communication events and employs a more representative dataset and feature-selection process, yielding slightly lower F1-scores for Facebook (0.89 vs. 1.0 in [20]) under substantially more realistic conditions.

Similarly, Li et al. [22] reported strong performance when identifying entry point (EP) methods in Twitter traffic, achieving an F1-score of 0.968 on a small dataset of 1706 flows. Nevertheless, their dataset was synthetically generated using the Monkey tool, which produced pseudorandom event sequences rather than actual user interactions. Their results, while high, therefore reflect an idealized setting rather than realistic communication patterns. Our work extends this line of research by analysing both message sending and reception across nine IM platforms using traffic generated through controlled, user-driven experiments. This design captures genuine behavioural variability and ensures reproducibility.

Study [21] classified instant messaging activities based on general usage types (e.g., text or image exchange) without distinguishing between sent and received messages. Their reported performance, estimated from graphical results, showed F1-scores of approximately 0.925 for WeChat and 0.975 for WhatsApp. In contrast, our approach introduces systematic feature selection and model optimization to explicitly discriminate message direction, providing finer granularity and improved interpretability of user actions.

To provide a controlled comparison, we additionally implemented the framework described in [21] using our WhatsApp dataset, as it is the platform with best performance in both studies. This framework was selected because it is the most closely related to our approach, as it employs hierarchical traffic segmentation into flow, session, and dialogue levels combined with a Random Forest classifier. To implement this framework, a Python script was developed to process the PCAP files, group packets into connection flows, and subsequently divide them into sessions, which were further segmented into dialogs. Both segmentation steps followed a time-interval criterion between packets, as described in the original work. In addition, dialogs containing a very small number of packets were discarded before training and testing a Random Forest model with 100 trees.

The evaluation was conducted under the same experimental configuration used in this study, including a 50% training and 50% testing split. However, the original work only addressed the detection of general usage types and did not distinguish between sent and received messages. In our implementation, we used our dataset, which includes distinctions between multimedia and text messages as well as between sent and received messages (see Table 4).

Under these identical experimental conditions, the reproduced results achieved precision, recall, and F1-score of 0.87. In comparison, our proposed framework achieves precision, recall, and F1-scores of approximately 0.98 on the WhatsApp dataset, demonstrating an improvement of 0.1 points thanks to the effectiveness of the proposed transaction-centric segmentation and feature selection strategy in accurately identifying user activities in encrypted instant messaging traffic.

In [23], user activity detection was evaluated on multiple applications, including Snapchat and WhatsApp, with the goal of identifying generic chat sessions rather than specific message events. Our method extends this scope by identifying message direction and type, offering a more detailed behavioural characterization. This refinement translates into higher accuracy, with F1-score improvements of approximately 0.3 points for Snapchat and 0.2 for WhatsApp. Although [23] also analysed Facebook activity, none of the events involved message transmission, underscoring the narrower scope of their study.

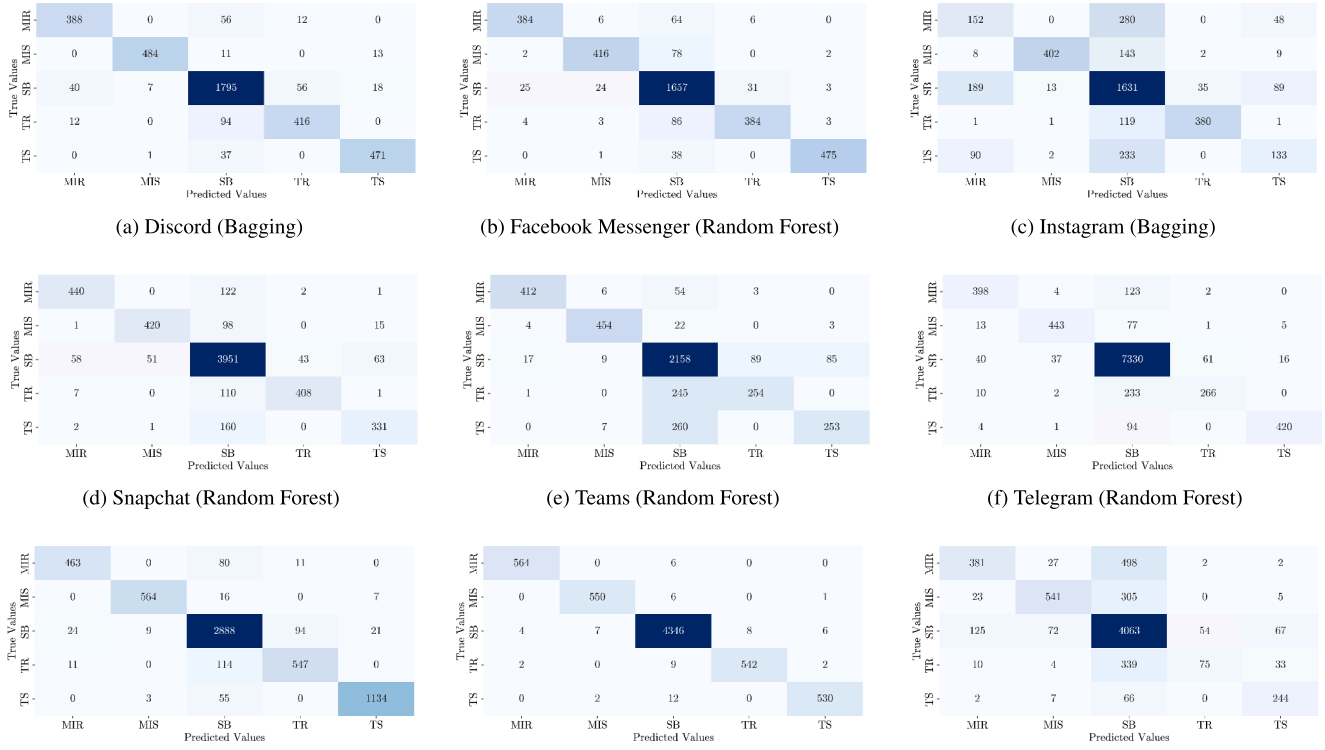
**Table 8**
Performance metrics obtained by the 3 best techniques and the MLP model across each platform.

| Platform | Model | Description | Prec | Recall | F1 | Acc | Error | Platform | Model | Description | Prec | Recall | F1 | Acc | Error |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Discord | Random Forest | n_estimators: 117, min_samples_split: 5, min_samples_leaf: 2, max_features: sqrt, max_depth: 10, bootstrap: False | 0.92 | 0.88 | 0.90 | 0.91 | 0.08 | Facebook Messenger | **Random Forest** | n_estimators: 69, min_samples_split: 5, min_samples_leaf: 2, max_features: sqrt, max_depth: 20, bootstrap: True | 0.92 | 0.87 | **0.89** | 0.90 | 0.09 |
| | **Bagging** | n_estimators: 101, max_features: 1, boostrap_features: True, boostrap: True | 0.91 | 0.89 | **0.90** | 0.91 | 0.08 | | Bagging | n_estimators: 113, max_features: 0.7, boostrap_features: True, boostrap: True | 0.91 | 0.86 | 0.88 | 0.89 | 0.10 |
| | Boosting | n_estimators: 133, learning_rate: 0.1 | 0.87 | 0.87 | 0.87 | 0.88 | 0.11 | | Boosting | n_estimators: 21, learning_rate: 0.1 | 0.84 | 0.85 | 0.85 | 0.85 | 0.14 |
| | MLP | solver: adam, learning_rate: constant, hidden_layer_size: (32, 32), alpha: 0.0001, activation: tanh | 0.87 | 0.86 | 0.86 | 0.88 | 0.11 | | MLP | solver: adam, learning_rate: adaptive, hidden_layer_size: (32, 32), alpha: 0.05, activation: relu | 0.88 | 0.81 | 0.84 | 0.85 | 0.14 |
| Instagram | Random Forest | n_estimators: 69, min_samples_split: 5, min_samples_leaf: 2, max_features: sqrt, max_depth: 10, bootstrap: True | 0.72 | 0.59 | 0.62 | 0.72 | 0.27 | Snapchat | **Random Forest** | n_estimators: 69, min_samples_split: 2, min_samples_leaf: 2, max_features: sqrt, max_depth: 60, bootstrap: True | 0.87 | 0.80 | **0.83** | 0.88 | 0.11 |
| | **Bagging** | n_estimators: 85, max_features: 0.5, boostrap_features: True, boostrap: True | 0.68 | 0.62 | **0.64** | 0.70 | 0.29 | | Bagging | n_estimators: 117, max_features: 0.7, boostrap_features: True, boostrap: True | 0.87 | 0.78 | 0.82 | 0.88 | 0.11 |
| | Boosting | n_estimators: 133, learning_rate: 1 | 0.62 | 0.31 | 0.62 | 0.67 | 0.32 | | Boosting | n_estimators: 150, learning_rate: 10 | 0.77 | 0.76 | 0.77 | 0.84 | 0.15 |
| | MLP | solver: adam, learning_rate: constant, hidden_layer_size: (32, 32), alpha: 0.05, activation: tanh | 0.75 | 0.56 | 0.59 | 0.71 | 0.28 | | MLP | solver: adam, learning_rate: adaptive, hidden_layer_size: (32, 32), alpha: 0.05, activation: tanh | 0.74 | 0.67 | 0.70 | 0.81 | 0.18 |

**Table 8**
Continued.

| Platform | Model | Description | Prec | Recall | F1 | Acc | Error | Platform | Model | Description | Prec | Recall | F1 | Acc | Error |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Teams | **Random Forest** | n_estimators: 133, min_samples_split: 5, min_samples_leaf: 2, max_features: sqrt, max_depth: 60, bootstrap: True | 0.83 | 0.77 | **0.79** | 0.82 | 0.17 | Telegram | **Random Forest** | n_estimators: 85, min_samples_split: 5, min_samples_leaf: 2, max_features: sqrt, max_depth: 10, bootstrap: False | 0.90 | 0.77 | **0.82** | 0.92 | 0.07 |
| | Bagging | n_estimators: 117, max_features: 0.95, boostrap_features: True, boostrap: True | 0.82 | 0.76 | 0.79 | 0.81 | 0.18 | | Bagging | n_estimators: 117, max_features: 1, boostrap_features: True, boostrap: True | 0.89 | 0.77 | 0.82 | 0.92 | 0.07 |
| | Boosting | n_estimators: 85, learning_rate: 0.01 | 0.75 | 0.75 | 0.75 | 0.77 | 0.22 | | Boosting | n_estimators: 117, learning_rate: 10 | 0.75 | 0.78 | 0.76 | 0.88 | 0.11 |
| | MLP | solver: adam, learning_rate: constant, hidden_layer_size: (32, 32), alpha: 0.0001, activation: relu | 0.68 | 0.66 | 0.64 | 0.72 | 0.27 | | MLP | solver: adam, learning_rate: constant, hidden_layer_size: (32, 32), alpha: 0.05, activation: tanh | 0.83 | 0.71 | 0.76 | 0.90 | 0.09 |
| WeChat | Random Forest | n_estimators: 85, min_samples_split: 2, min_samples_leaf: 2, max_features: sqrt, max_depth: 20, bootstrap: True | 0.92 | 0.89 | 0.90 | 0.92 | 0.07 | WhatsApp | Random Forest | n_estimators: 100, min_samples_split: 10, min_samples_leaf: 4, max_features: sqrt, max_depth: 20, bootstrap: False | 0.98 | 0.98 | 0.98 | 0.98 | 0.01 |
| | **Bagging** | n_estimators: 117, max_features: 1, boostrap_features: True, boostrap: True | 0.92 | 0.90 | **0.91** | 0.92 | 0.07 | | **Bagging** | n_estimators: 101, max_features: 0.9, boostrap_features: True, boostrap: True | 0.98 | 0.98 | **0.98** | 0.98 | 0.01 |
| | Boosting | n_estimators: 101, learning_rate: 0.1 | 0.87 | 0.87 | 0.87 | 0.88 | 0.11 | | Boosting | n_estimators: 37, learning_rate: 0.001 | 0.97 | 0.98 | 0.97 | 0.98 | 0.01 |
| | MLP | solver: adam, learning_rate: adaptive, hidden_layer_size: (16, 32, 64), alpha: 0.0001, activation: relu | 0.84 | 0.84 | 0.84 | 0.87 | 0.12 | | MLP | solver: adam, learning_rate: adaptive, hidden_layer_size: (16, 32, 64), alpha: 0.05, activation: relu | 0.96 | 0.98 | 0.97 | 0.98 | 0.01 |
| X | **Random Forest** | n_estimators: 150, min_samples_split: 5, min_samples_leaf: 4, max_features: sqrt, max_depth: None, bootstrap: False | 0.73 | 0.58 | **0.62** | 0.77 | 0.22 | X | Boosting | n_estimators: 150, learning_rate: 0.1 | 0.57 | 0.57 | 0.57 | 0.69 | 0.30 |
| | Bagging | n_estimators: 185, max_features: 0.5, boostrap_features: True, boostrap: True | 0.71 | 0.55 | 0.59 | 0.75 | 0.24 | | MLP | solver: adam, learning_rate: constant, hidden_layer_size: (32, 32), alpha: 0.0001, activation: tanh | 0.68 | 0.50 | 0.54 | 0.72 | 0.27 |

**Table 9**
Confusion matrices.




(a) Discord (Bagging) (b) Facebook Messenger (Random Forest) (c) Instagram (Bagging)

(d) Snapchat (Random Forest) (e) Teams (Random Forest) (f) Telegram (Random Forest)

(g) WeChat (Bagging) (h) WhatsApp (Bagging) (i) X (Random Forest)

**Table 10**
Comparative performance of user activity detection methods extracted from the literature, alongside our corresponding results.

| Ref. | Platform | Event | Prec. | Recall | F1 | Our Prec. | Our Recall | Our F1 |
|---|---|---|---|---|---|---|---|---|
| [20] | Facebook | Send message | 1.00 | 1.00 | 1.00 | 0.92 | 0.87 | 0.89 |
| | Twitter | Tweet /message | 1.00 | 0.95 | 0.97 | 0.73 | 0.58 | 0.62 |
| [22] | Twitter | EP methods | 1.000 | 0.939 | 0.968 | 0.73 | 0.58 | 0.62 |
| [21] | WeChat | Usage | ~0.925 | ~0.95 | ~0.925 | 0.92 | 0.90 | 0.91 |
| | WhatsApp | Usage | ~0.95 | ~0.975 | ~0.975 | 0.98 | 0.98 | 0.98 |
| [23] | WhatsApp | Chat | 0.6471 | 1.00 | 0.7857 | 0.98 | 0.98 | 0.98 |
| | Snapchat | Chat | 0.4706 | 0.7273 | 0.5861 | 0.87 | 0.80 | 0.83 |

Collectively, these results reveal a clear trade-off between dataset realism and reported accuracy. Prior works achieved near-perfect metrics under constrained, synthetic, or platform-specific conditions, whereas our method attains competitive performance across nine heterogeneous IM services under real user-driven traffic. This demonstrates the scalability and robustness of the proposed framework across diverse network environments.

In summary, the proposed approach provides higher cross-platform generalizability, a transparent feature-engineering pipeline, and the capacity to distinguish between message sending and receiving with event-level precision. These characteristics, together with the use of realistic datasets, establish it as a more interpretable and practically applicable framework for analysing encrypted instant messaging traffic, advancing the state of the art toward comprehensive and privacy-aware user activity detection.

## 5. Ethical and privacy considerations

Activity fingerprinting has the potential to reveal behavioural patterns even when message content remains encrypted. For this reason, all experiments in this study were conducted under strict ethical and privacy safeguards. The dataset was generated entirely within controlled laboratory environments, and no personal or identifiable information was captured or analysed. Only flow-level metadata such as packet size, direction, and timing were processed, while payloads were excluded at all times. All traffic was anonymized immediately after capture, and the experiments complied with institutional research ethics guidelines and the principles of the General Data Protection Regulation (GDPR). The proposed framework is designed for legitimate, consent-based applications, such as network forensics, intrusion detection, and system performance analysis, where visibility into

encrypted traffic patterns can provide valuable diagnostic or security insights.

We are aware, however, that the same techniques could be misused for unauthorized surveillance or profiling. To mitigate this risk, we explicitly discourage any application of this work outside lawful and ethical boundaries and emphasize the need for transparent oversight and informed consent when such analyses are performed. Our results also reveal potential side-channel vulnerabilities in the way some instant messaging platforms manage their encrypted traffic. Therefore, we discuss possible countermeasures, such as traffic padding, batching, or timing randomization, that can be implemented by service providers to reduce the information leakage exploited by fingerprinting techniques. By addressing these aspects, we aim to contribute not only to the technical advancement of encrypted traffic analysis but also to its responsible and ethical deployment in real-world contexts.

## 6. Conclusion

The fingerprinting of message transmissions, both text and multimedia, can be achieved with a high degree of precision across most instant messaging platforms. Our results demonstrate F1 scores ranging from 0.98 for WhatsApp to 0.62 for X. Based on these findings, the models can be confidently applied to platforms such as WhatsApp, where they exhibit near-perfect accuracy, while for platforms like X, their reliability may be insufficient for confident deployment.

Overall, the proposed methodology, encompassing empirical data generation, connection characterisation, feature extraction and selection, and machine learning-based classification, proved effective for accurate and real-time fingerprinting of encrypted instant messaging activities. By leveraging transaction-level analysis instead of complete connections, the framework supports real-time detection with low latency and high interpretability.

Beyond its technical contribution, this work provides an empirical foundation for understanding how encrypted traffic can inadvertently expose user activity patterns. The findings highlight both the potential of such analysis for legitimate security and forensic applications, and the need for responsible, privacy-conscious deployment. To mitigate the risks of activity fingerprinting, messaging platforms should consider implementing countermeasures such as traffic padding, batching, or timing randomisation to disrupt identifiable patterns. Future work may explore adaptive defences and the transferability of trained models across evolving platform architectures.

## CRediT authorship contribution statement

**Lorena Mehavilla:** Writing – review & editing, Validation, Software, Methodology, Investigation, Conceptualization; **José García:** Writing – review & editing, Validation, Software, Methodology, Investigation, Conceptualization; **Álvaro Alesanco:** Writing – review & editing, Validation, Software, Methodology, Investigation, Conceptualization.

## Data availability

Data will be made available on request.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgment

This research is funded by Government of Aragón [reference group Cenit T31_20R]. In addition, the publication is part of the proyect PID2022-136476OB-I00 funded by MICIU/AEI/10.13039/501100011033/FEDER,UE and Lorena Mehavilla was supported in part by the Government of Aragón [grant for predoctoral research contracts (2023–2027)].

## References

[1] F. Duarte, Exploding Topics, Most Popular Messaging Apps, 2024. https://explodingtopics.com/blog/messaging-apps-stats,26/06/2024.
[2] Blog, Most Popular Messaging Apps Worldwide, 2023. https://www.similarweb.com/blog/research/market-research/worldwide-messaging-apps/,31/01/2024.
[3] A. Bahramali, R. Soltani, Houmansadr, Amir, D. Goeckel, D. Towsley, Practical Traffic Analysis Attacks on Secure Messaging Applications, 2020.
[4] M. Conti, L.V. Mancini, R. Spolaor, N.V. Verde, Analyzing Android Encrypted Network Traffic to Identify User Actions, 11, 2016. https://doi.org/10.1109/tifs.2015.2478741
[5] C. Hou, J. Shi, C. Kang, Z. Cao, X. Gang, Classifying user activities in the encrypted WeChat traffic, in: IEEE 37th International Performance Computing and Communications Conference (IPCCC), Orlando, FL, USA, 2018, pp. 1–8. https://doi.org/10.1109/PCCC.2018.8711267
[6] M. Shen, Machine learning-powered encrypted network traffic analysis: a comprehensive survey, IEEE Commun. Surv. Tutorials 25 (1) (2023) 791–824. https://doi.org/10.1109/COMST.2022.3208196
[7] T.V. Ede, R. Bortolameotti, A. Continella, J. Ren, D.J. Dubois, M. Lindorfer, D.R. Choffnes, M.V. Steen, A. Peter, Semi-Supervised Mobile-App Fingerprinting on Encrypted Network Traffic. Proceedings 2020 Network and Distributed System Security Symposium, Technical Report, FlowPrint, 2020.
[8] J. Li, S. Wu, H. Zhou, X. Luo, T. Wang, Y. Liu, X. Ma, Packet-level open-world app fingerprinting on wireless traffic, in: Proceedings of the 29th Network and Distributed System Security Symposium (NDSS), the 29th Network and Distributed System Security Symposium (NDSS), 2022, pp. 1–18.
[9] V.F. Taylor, R. Spolaor, M. Conti, I. Martinovic, Robust smartphone app identification via encrypted network traffic analysis, IEEE Trans. Inf. Forensics Secur. 13 (2018) 63–78. https://doi.org/10.1109/TIFS.2017.2737970
[10] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, J. Aguilar, Towards the deployment of machine learning solutions in network traffic classification: a systematic survey, IEEE Commun. Surv. Tutorials 21 (2) (1988). https://doi.org/10.1109/COMST.2018.2883147
[11] L. Ceci, Statista, Number of Registered Discord Users Worldwide From, 2017. https://www.statista.com/statistics/1367922/discord-registered-users-worldwide/,01/08/2023.
[12] J. Stacy, S. Dixon, Number of Instagram Users Worldwide From 2020 to 2025, 2025. https://www.statista.com/statistics/183585/instagram-number-of-global-users/,22/05/2024.
[13] R. Shewale, Usage & Revenue, Microsoft Teams Statistics, 2024. https://www.demandsage.com/microsoft-teams-statistics/,14/03/2024.
[14] G. Jay, X. (formerly Twitter, Usage Statistics For, 2024. https://famewall.io/statistics/twitter-stats/,05/08/2024.
[15] C. Anglano, Forensic Analysis of WhatsApp Messenger on Android Smartphones, Digital Investigation, 11, 2014. https://doi.org/10.1016/j.diin.2014.04.003
[16] K. Kaushik, Y. Katara, Forensic analysis of WhatsApp chat data, in: 2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2022, pp. 1–6. https://doi.org/10.1109/ICRITO56286.2022.9965028
[17] P. Fiadino, Schiavone, Mirko, P. Casas, Vivisecting WhatsApp in Cellular Networks: Servers, Flows, and Quality of Experience, 2015, 49–63. https://doi.org/10.1007/978-3-319-17172-2_4
[18] S. Ramraj, G. Usha, Signature identification and user activity analysis on WhatsApp web through network data, Microprocess. Microsyst. 97 (2023). https://doi.org/10.1016/j.micpro.2023.104756
[19] S. Abd, E. Sarhan, H.A. Youness, M. Ayman, Bahaa-Eldin, A framework for digital forensics of encrypted real-time network traffic, instant messaging, and VoIP application case study, Ain Shams Eng. J. 14 (9) (2023) 102069. https://doi.org/10.1016/j.asej.2022.102069
[20] M. Conti, L.V. Mancini, R. Spolaor, N.V. Verde, Analyzing android encrypted network traffic to identify user actions, IEEE Trans. Inf. Forensics Secur. 11 (2016) 114–125. https://doi.org/10.1109/TIFS.2015.2478741
[21] Y. Fu, H. Xiong, X. Lu, J. Yang, C. Chen, Service usage classification with encrypted internet traffic in mobile messaging apps, IEEE Trans. Mob. Comput. 15 (2016) 2851–2864. https://doi.org/10.1109/TMC.2016.2516020

[22] J. Li, H. Zhou, S. Wu, X. Luo, T. Wang, X. Zhan, X. Ma, FOAP: Fine-Grained Open-World Android App Fingerprinting, 31st USENIX Security Symposium (USENIX Security 22), Boston, MA, https://www.usenix.org/conference/usenixsecurity22/presentation/li-jianfeng.

[23] B. Saltaformaggio, H. Choi, K. Johnson, Y. Kwon, Q. Zhang, X. Zhang, D. Xu, J. Qian, Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic, in: Proceedings of the 10th USENIX Conference on Offensive Technologies (WOOT'16), the 10th USENIX Conference on Offensive Technologies (WOOT'16), USENIX Association, USA, 2016, pp. 69–78.

[24] Y.X. Meng, The practice on using machine learning for network anomaly intrusion detection, in: 2011 International Conference on Machine Learning and Cybernetics, Guilin, China, 2011, pp. 576–581. https://doi.org/10.1109/ICMLC.2011.6016798

[25] N. Saran, N. Kesswani, A comparative study of supervised machine learning classifiers for intrusion detection in internet of things, Procedia Comput. Sci. 218 (2023) 2049–2057. https://doi.org/10.1016/j.procs.2023.01.181

[26] J. Bhayo, A. Syed, S. Shah, A. Hameed, J. Ahmed, D. Nasir, Draheim, Towards a Machine Learning-Based Framework for DDOS Attack Detection in Software-Defined IoT (SD-IoT) Networks, Engineering Applications of Artificial Intelligence, (2023), 123, Part C., 106432, https://doi.org/10.1016/j.engappai.2023.106432

[27] Zeek Network Security Monitor, https://zeek.org/.