Mobile Application Identification in Encrypted Traffic Using JA4+ Fingerprints

Marta Fernández-Terrasa, Jorge E. López de Vergara, Francisco J. Gómez-Arribas, Luis de Pedro, and Iván González

Department of Electronic and Communication Technologies, School of Engineering, Universidad Autónoma de Madrid, Spain
marta.fernandezt@estudiante.uam.es, {jorge.lopez_vergara, francisco.gomez, luis.depedro, ivan.gonzalez}@uam.es

Abstract. The increasing adoption of TLS 1.3 and the Encrypted Client Hello (ECH) extension poses significant challenges to network traffic classification, as traditional identifiers like the Server Name Indication (SNI), which provides the server domain name, are no longer visible. In this work, we explore the use of JA4+, a suite of TLS and TCP fingerprinting techniques, to identify mobile applications without relying on SNI data. We conduct a large-scale experimental study involving 50 mobile applications across 11 real devices, capturing network traffic and extracting high-entropy fingerprints. We evaluate three classification strategies—two machine learning models and a dictionary-based approach—and assess their performance using multiple variants of JA4+ fingerprints. Our findings show that the dictionary-based model achieves the highest accuracy and lowest classification time, making it suitable for practical deployment in privacy-preserving network environments. Unlike prior works that rely on exposed domain names, such as those using SNI, our approach remains effective even under fully encrypted TLS handshakes. This makes JA4+ a promising tool for secure, scalable, and future-proof encrypted traffic analysis.

Keywords: TLS 1.3 · JA4+ · Encrypted Traffic · ECH · Mobile App Fingerprinting · Privacy-preserving Classification

1 Introduction

In recent years, the volume and diversity of encrypted network traffic have increased drastically. With users relying more than ever on mobile devices for tasks ranging from communication and banking to navigation and entertainment, ensuring privacy and security in data transmission has become paramount. This has led to the widespread deployment of Transport Layer Security (TLS) 1.3 [11], which improves confidentiality by encrypting a larger portion of the handshake protocol.

A crucial innovation in TLS 1.3 is the Encrypted Client Hello (ECH) extension [12]. ECH encrypts the Server Name Indication (SNI) field, which was previously visible during the initial handshake and often used for identifying the target server or hosted application, as it provided the domain name of the server in plaintext. The encryption of this field eliminates traditional mechanisms used by ISPs, firewalls, and network operators to classify and manage traffic.

While this shift enhances privacy and security, it also presents significant challenges for researchers and practitioners who rely on metadata-based traffic analysis. Legacy techniques that depend on exposed SNI values or observable packet contents are not viable any longer. As a result, new approaches are needed to maintain visibility into encrypted network traffic, particularly for purposes such as security monitoring, policy enforcement, and traffic optimization.

In this work, we focus on fingerprinting techniques based on JA4+ [6], a suite of TLS and transport-layer fingerprints that capture behavioral characteristics of a connection. Unlike previous techniques such as JA3 [2], which offered a relatively coarse and limited view of the TLS handshake, JA4+ provides a more detailed, extensible, and structured format for characterizing clients and servers.

The main objective of this study is to assess whether JA4+ fingerprints can be reliably used to identify mobile applications, even in the presence of ECH. We explore this by building a custom dataset using real mobile devices and analyzing traffic generated by 50 popular apps across different categories. We implement three different classification strategies and evaluate their performance based on accuracy, robustness, and execution time.

Our goal is to design a solution that is both effective—in terms of high identification accuracy—and efficient—with low classification time, suitable for real-time deployment. By evaluating several classification strategies and fingerprint variants, we demonstrate that JA4+ enables scalable and accurate encrypted traffic analysis, even in fully encrypted scenarios.

Despite relying on publicly available fingerprinting formats, this study offers a novel empirical perspective by validating JA4+ under realistic, privacy-preserving TLS settings where SNI is fully encrypted. Unlike prior works that assume access to domain names or operate in controlled environments, we conduct live traffic capture from real user devices without root access or instrumentation, and evaluate classification performance across platforms and protocols. This hands-on evaluation closes a critical gap between theoretical fingerprinting capabilities and their practical deployment, positioning JA4+ as a practical, scalable, and effective baseline for encrypted traffic classification in real-world scenarios.

The remainder of this paper is structured as follows: Section 2 reviews the existing literature and contrasts our approach with related work. Section 3 explains the JA4+ fingerprinting. Section 4 presents the methodology and details the experimental setup. Section 5 details the classification techniques used and the experimental protocol. Section 6 provides performance results, and a discussion is later given in Section 7. Finally, Section 8 summarizes our findings, and outlines directions for future research.

2 Related Work

TLS fingerprinting has been studied extensively over the years [3], particularly through the use of techniques like JA3 and its extensions. JA3 [2] was first introduced by Salesforce as a method for identifying TLS clients based on the ordered list of cipher suites and extensions advertised in the ClientHello message. While effective in some environments, JA3 has been shown to suffer from significant limitations, collision of finger-

prints across unrelated applications and high sensitivity to client-side implementation details, including randomly ordered cipher suites.

Recent advances have shifted toward more robust and extensible fingerprinting formats. The JA4+ family, developed by FoxIO [6], addresses many of JA3 limitations by expanding the fingerprinting scope to include transport-layer features, server-side parameters (JA4S), certificate characteristics (JA4X), and TCP fields and options (JA4TS). This provides a higher-entropy representation of client-server behaviors.

One of the most relevant works in this area is the study by the University of Brno, including their recent 2024 publication [4]. In that paper, the authors analyze the reliability of JA4 fingerprints for identifying applications in encrypted traffic. Their approach demonstrates good performance by combining JA4 data with the Server Name Indication (SNI), which remains exposed in their dataset. They argue that the inclusion of SNI significantly improves classification precision and recall, especially for applications that share similar TLS behavior.

Our approach diverges from this in a crucial way: we assume a realistic deployment scenario where ECH is active and SNI is not accessible. This makes our classification task substantially more difficult, but also more aligned with modern privacy-preserving TLS configurations. Rather than relying on domain name leakage, our method seeks to extract maximum information from observable protocol behaviors alone.

In addition, while the researchers at University of Brno employed conventional dictionary-matching on static fingerprints and emphasized on visualization and taxonomy, our work introduces an inverted dictionary model that explicitly supports fingerprint overlap and ambiguity. We also benchmark two machine learning classifiers to contextualize the trade-offs of heuristic versus learned approaches.

Other studies, such as [1], have used statistical features (e.g., packet size and inter-arrival time) for mobile app identification. While promising, these methods are difficult to maintain in the presence of transport protocol changes, and are not easily generalized across devices. Our use of JA4+ avoids this by staying within protocol-compliant feature extraction.

Overall, our work contributes a fully SNI-independent, real-device, multi-platform evaluation of JA4+ fingerprinting for mobile applications—filling a gap in current literature.

3 JA4+ Fingerprinting

JA4+ [6] is an evolution of JA3, incorporating features from both client and server sides of the TLS handshake, as well as X.509 certificate and TCP characteristics. It includes, among others:

- **JA4 (*ClientHello*)**: Encodes cipher suites, TLS version, ALPN (Application-Layer Protocol Negotiation), and transport protocol.
- **JA4S (*ServerHello*)**: Captures server TLS configuration and negotiated parameters.
- **JA4X (X.509 Certificate)**: Represents the structure and fields of TLS certificates (it focuses on how the certificates have been generated, not their content).
- **JA4TS (TCP SYN-ACK)**: Extracts server response behavior at the TCP layer.

This multi-layer design provides a broader behavioral view, increasing fingerprint entropy and reducing ambiguity.

4 Methodology and Experimental Setup

To evaluate the practical applicability of JA4+ fingerprints for identifying mobile applications, we designed a comprehensive experimental framework encompassing traffic generation, data capture, feature extraction, and classification, as shown in Figure 1. This section describes each component in detail.

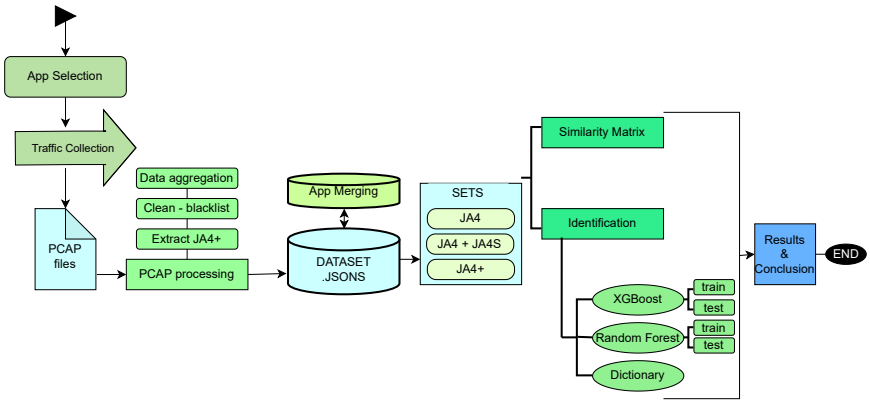

Fig. 1. Experimental framework

4.1 Device and Mobile Application Selection

Our experimental dataset includes traffic from 50 popular mobile applications, selected to span a diverse range of categories such as social media, video streaming, communication, banking, and navigation. The app list was curated based on app store rankings and usage statistics in Spain during early 2025. This selection ensures representativeness of the modern mobile ecosystem.

The list of applications, ordered alphabetically, is: Adobe Acrobat Reader, AliExpress, Amazon, Banco Santander, BBVA, BlaBlaCar, Bolt, Brave Private Web Browser, Burger King, Cabify, CaixaBankNow, CapCut, ChatGPT, Discord, Disney+, Facebook, Glovo, Gmail, Google, Google Chrome, Google Drive, Google Maps, Google Translate, Imagin, Instagram, Just Eat, Max (HBO Max), Messenger, Microsoft Teams, Moovit, Netflix, Pinterest, Prime Video, Renfe Cercanías, Revolut, SHEIN, Snapchat, Spotify, Telepark, Telepizza, Temu, Threads, TikTok, Uber, UberEats, Wallapop, Waze, WhatsApp, YouTube, Zoom.

Figure 2 shows a histogram of the number of JA4 fingerprints per mobile application. As seen, the number of fingerprints is very variable, being between 300 and 600

the most common value, but ranging from 100 to 2200. This variability depends on the complexity of the app, since some receive a higher volume of traffic and others could not be installed on all devices.

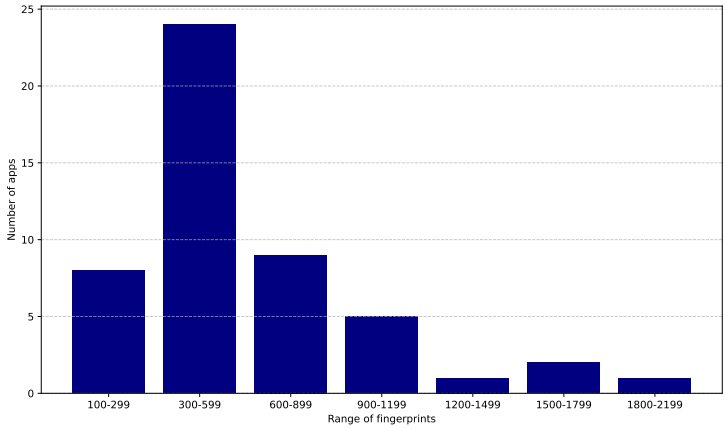

Fig. 2. Distribution of JA4 fingerprint per application

Traffic was collected from 11 physical mobile devices: 3 Android smartphones running Android 11 or MIUI 12, and 8 iPhones operating on iOS versions ranging from 16 to 18. These devices were sourced from different users to introduce natural variability in configuration, usage history, and system background activity. No additional instrumentation was installed to minimize behavioral drift from typical user activity. There was no notable difference in the number of fingerprints collected across devices, except two mobile phones, where not all applications could be captured due to installation limitations.

4.2 Network and Capture Infrastructure

To collect TLS traffic without modifying the devices, we configured a Windows 11 laptop as a Wi-Fi access point. The mobile devices were connected to this AP, and all their network traffic was routed through the laptop. This setup enabled transparent traffic monitoring without requiring root access or packet injection on the mobile devices. Since the dataset was collected over Wi-Fi rather than cellular networks, latency and congestion patterns typical of mobile networks may not be fully reflected in the dataset. Nevertheless, this is not a real problem for JA4+ fingerprints, which are largely independent of the underlying physical link.

Wireshark (v4.2) [13] was used to capture packets in PCAPNG format. The system had the JA4 plugin from FoxIO installed, allowing in-line fingerprint extraction for JA4, JA4S, JA4X, and JA4TS. Each app was launched multiple times with varied user interaction levels, ranging from idle opening to active navigation and content consumption, to capture a wide range of behavioral patterns.

4.3 Traffic Processing and Fingerprint Extraction

The captured traffic was processed in several stages:

1. **Filtering:** Non-application-related traffic (e.g., OS updates, background services) was filtered using a combination of the OISD blocklist [8], Peter Lowe's list [7], and a curated set of 46 known Apple and Google service domains (e.g.: play.googleapis.com or itunes.apple.com).
2. **JA4+ Feature Extraction:** For each TCP/TLS flow, we extracted the following:
   - *JA4:* Characteristics of the TLS *ClientHello* record.
   - *JA4S:* Features from the *ServerHello* record.
   - *JA4X:* Encoded structure of the X.509 certificate.
   - *JA4TS:* Server TCP behavior from SYN-ACK responses.
3. **Aggregation:** Fingerprints from a session were aggregated into per-app fingerprint sets. These were exported to JSON files grouped by app and platform.

4.4 Fingerprint Variants

To study the relative contribution of each component, we constructed four fingerprint variants:

- JA4 only.
- JA4 + JA4S.
- JA4 + JA4S + JA4TS.
- Full JA4+ suite as JA4 + JA4S + JA4TS + JA4X.

This decomposition allowed us to analyze which parts of the TLS and TCP handshake provide the most discriminatory power for app identification.

5 Classification Models and Experimental Protocol

This section describes the design and implementation of the classification models used to evaluate the utility of JA4+ fingerprints in encrypted mobile app identification. We describe the data representation techniques, the models evaluated, and the experimental protocols applied.

5.1 Feature Representation and Preprocessing

JA4+ fingerprints are structured textual identifiers capturing different layers of connection metadata. To prepare these for machine learning classification, each fingerprint was converted into a feature vector using Scikit-learn `FeatureHasher`. This technique converts textual tokens into fixed-length binary vectors (of size 200) using a hashing trick that preserves sparsity and co-occurrence patterns.

We treated each fingerprint string (JA4, JA4S, JA4X, JA4TS) as a categorical feature. When using composite variants (e.g., JA4 + JA4S), the components were concatenated using delimiters to preserve structure.

5.2 Machine Learning Classifiers

We evaluated two supervised learning algorithms:

- **Random Forest:** An ensemble of 150 decision trees using Gini impurity, suitable for handling sparse binary data. Random Forests are known for their robustness to overfitting and noise.
- **XGBoost:** An implementation of gradient boosting over decision trees, optimized for speed and accuracy.

Both models were trained using 15-fold stratified cross-validation. This ensured that each app appeared in both training and validation folds in proportion to its frequency. Importantly, we avoided any leakage of closely related samples across folds: each fingerprint is uniquely assigned to a fold, and multiple fingerprints from the same TCP connection or session are never split between training and test sets. This setup guarantees that the classifier performance reflects its ability to generalize to new flows.

5.3 Dictionary-Based Fingerprint Matching

We also implemented a heuristic model based on dictionary matching. In this approach, all fingerprints from the training set were stored in an inverted index mapping fingerprints to their corresponding applications. During inference, a new fingerprint was queried against the dictionary:

- If an exact match was found, the model returned the associated apps ranked by frequency.
- If multiple apps shared the fingerprint, a voting scheme was applied.
- If no match existed, the app was classified as unknown.

This model is computationally efficient and well-suited for use cases requiring real-time classification. While it cannot generalize beyond observed fingerprints, its accuracy is high when dealing with high-entropy inputs like JA4+.

5.4 Evaluation Protocol

Each model was evaluated on multiple fingerprint variants to determine how different layers (TLS handshake, server response, certificate info, and TCP behavior) contribute to app identification accuracy. Metrics used include:

- **Top-k Accuracy:** Percentage of samples where the correct app appeared among the first k predictions. We measure the results for $k \in \{1, 3, 5\}$.
- **Confusion Matrices:** Used to visualize app-level misclassification patterns.
- **Inference Time:** Average time (in milliseconds) to classify one fingerprint.

This evaluation setup allows for both performance comparison across models and analysis of fingerprint expressiveness.

6 Results and Evaluation

In this section, we present the performance results of the classification methods introduced previously. The analysis covers accuracy, runtime efficiency, and robustness across different fingerprint variants and platforms.

6.1 Intra-Model Accuracy Across Fingerprint Sets

To better understand how each type of fingerprint contributes to classification accuracy, we conducted a detailed intra-model evaluation. In this experiment, we trained separate dictionary-based models using different fingerprint combinations (JA4, JA4+JA4S, JA4+JA4S+JA4TS, and JA4+JA4S+JA4TS+JA4X) and measured their classification performance.

Figure 3 illustrates the results. Given the 50-class setup, a random classifier would yield a baseline accuracy of 2%. We observe that accuracy increases progressively as more fingerprint components are included, confirming the value of combining orthogonal fingerprint sources. Moreover, we see that the dictionary-based approach shows better figures than the machine learning-based classifiers. In this case, the top-1 accuracy increases from 31.72% with JA4 only, to 42.91% with JA4+JA4S, then to 57.12% with the inclusion of JA4TS, and finally peaks at 58.49% with the full JA4+ fingerprint suite.

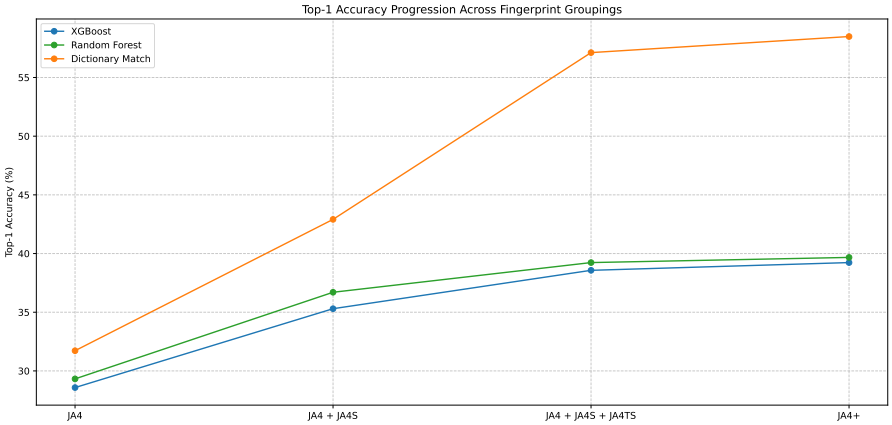

Fig. 3. Top-1 accuracy of the dictionary model trained on individual JA4+ fingerprint sets.

This substantial improvement indicates that the model is highly sensitive to fingerprint uniqueness, gaining much more from added characteristics in the fingerprints than the machine learning-based classifiers. This is because this approach performs exact matches rather than generalizing, so each new field added helps to reduce overlap and ambiguity with other mobile applications.

These results confirm that no single fingerprint dominates all others in isolation, and support the hypothesis that their combination is beneficial. The components are partially orthogonal in the information they capture: while JA4 reflects client configuration, JA4S captures negotiated parameters, JA4TS expresses server low-level network stack behavior, and JA4X encodes structural identity. Together, they form a richer representation that enhances overall classification accuracy.

6.2 Top-k Accuracy

Figure 4 summarizes the top-1, top-3, and top-5 accuracy for each classification model using the full JA4+ fingerprint set. The dictionary-based method achieved the highest performance overall: 58.5% for top-1 accuracy, 77.9% for top-3 accuracy, and 88.4% for top-5 accuracy.

In contrast, Random Forest and XGBoost models reached a top-1 accuracy of approximately 39.6% and 38.4%, respectively, with their top-5 performance lagging behind the dictionary model by 8–10 percentage points.

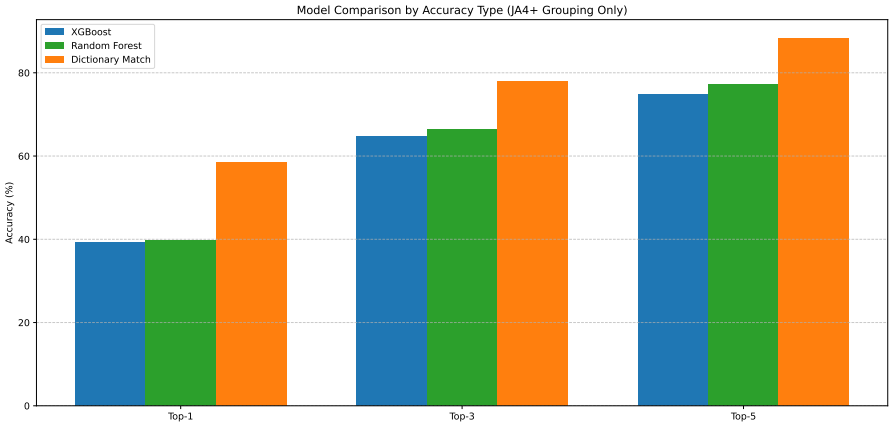

Fig. 4. Top-k accuracy across classifiers using full JA4+ fingerprinting.

6.3 Platform-specific Accuracy

When disaggregating results by mobile platform, we observed slightly better performance on Android devices compared to iOS, as shown in Table 1. This may be attributed to platform differences in TLS stack behavior.

6.4 Inference Time Analysis

Figure 5 compares the average inference time per sample across the three classifiers using a commodity laptop (Intel Core i7-8565U CPU @ 1.80GHz, 4 cores, 8 threads, 20

Table 1. Classifier accuracy by platform using dictionary-based model.

Platform	Top-1	Top-3	Top-5
Android	58.3%	77.6%	83.4%
iOS	54.4%	72.5%	81.0%

GB RAM). The dictionary-based model clearly outperforms the learning-based models in execution time, with 0.028 ms/sample, compared to the 0.208 ms/sample and 0.221 ms/sample of Random Forest and XGBoost, respectively.

This efficiency of the dictionary-based model, combined with the absence of a training phase and low memory requirements, makes it particularly attractive for high-throughput environments such as network middleboxes or mobile gateways, being able to deal with tens of thousands flows per second and processor core.

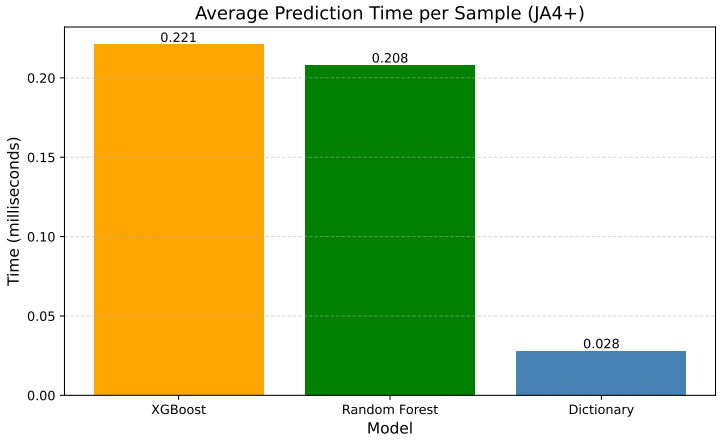

Fig. 5. Average inference time per sample.

6.5 Confusion and Ambiguity Analysis

To better understand the misclassifications, we examined confusion matrices for each classifier, shown in Figure 6. We found that most misclassifications occurred among functionally or commercially related applications. For instance, Amazon and Prime Video shared a significant number of fingerprints due to common backend infrastructure. Similar overlaps were observed between CapCut and TikTok, both owned by ByteDance.

Additionally, many of these overlaps stem from shared infrastructure at the network level. Several applications rely on common content delivery networks (CDNs), authentication providers, or embedded SDKs—such as Firebase, Google APIs, Cloudflare or

Akamai—which can lead to identical or near-identical JA4+ fingerprints, especially when TLS configurations and stacks are uniform across apps. This represents a fundamental limitation of passive fingerprinting techniques that rely solely on protocol-level metadata.

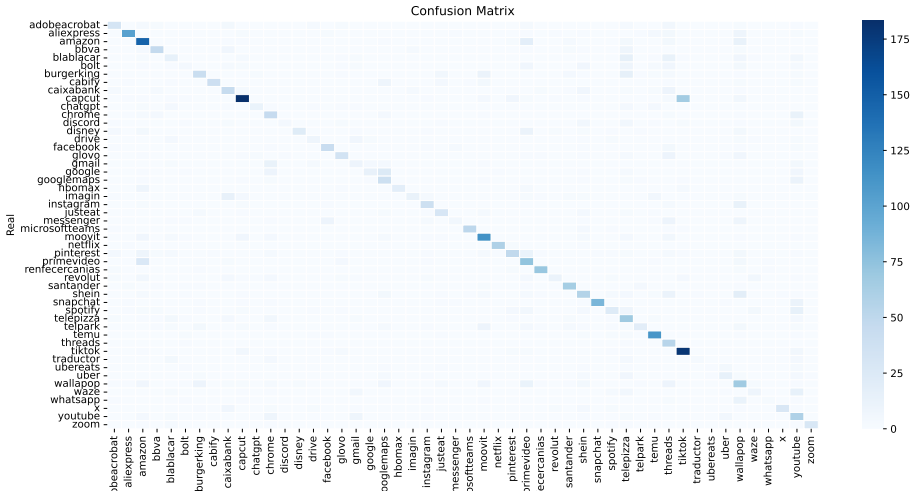

Fig. 6. Confusion Matrix among different apps with JA4+ fingerprints and dictionary classifier

Additionally, apps using system webviews or SDK-based integrations (e.g., ad libraries) tended to produce non-unique fingerprints, increasing ambiguity. This insight suggests a possible direction for future work: incorporating session-level behavioral metrics or leveraging temporal fingerprint sequences.

Moreover, we also generated a similarity matrix to see the number of JA4+ fingerprints shared among different applications, as shown in Figure 7. Although it suggest a better result should be obtained, it also shows many shared fingerprints among all applications, which are likely to contribute to classification ambiguity.

6.6 Generalization and Collision Handling

All reported metrics reflect performance under stratified 15-fold cross-validation, where no fingerprints from the same session are shared between training and test folds. This setup enforces a strict evaluation regime and supports the interpretation that the models generalize to new, unseen flows. Despite the use of hashing (via FeatureHasher) to vectorize categorical fingerprint components, no adverse effects were observed in terms of classification accuracy or consistency. We did not detect significant collisions that impaired model performance, likely due to the limited dimensionality of each JA4 component and the deterministic behavior of the hashing.

However, we note that the dictionary-based approach, while accurate and efficient, cannot classify previously unseen fingerprints. If a JA4+ tuple does not exactly match

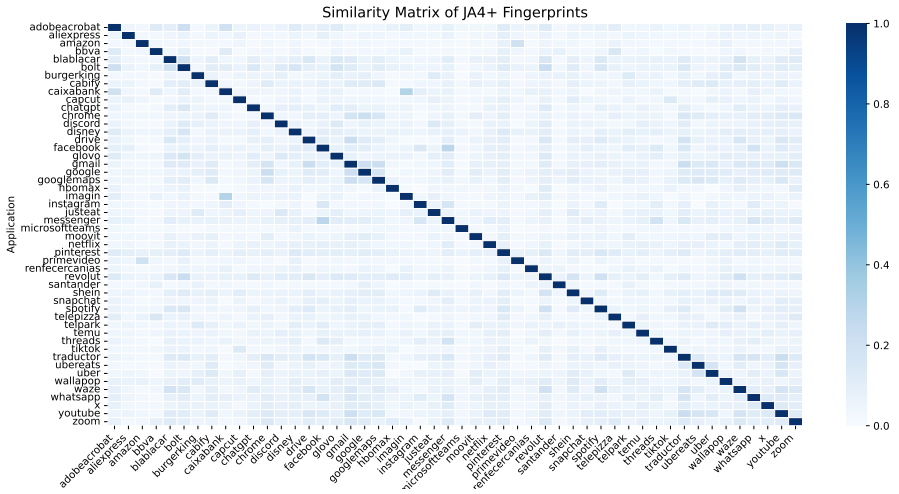

Fig. 7. Similarity matrix showing the JA4+ fingerprints of different applications

any entry in the dictionary, the app is classified as unknown. This is a fundamental limitation of the method that could be addressed in future work by hybridizing with statistics-based models.

6.7 Case Studies and Limit Scenarios

In addition to aggregate performance metrics, we conducted a qualitative analysis of cases where classification results were ambiguous or incorrect. These boundary scenarios help highlight the practical challenges and limits of JA4+ fingerprinting.

Shared Infrastructure: Some misclassifications stemmed from applications sharing backend infrastructure, as shown above. For instance, Amazon and Prime Video both rely on Amazon Web Services and Content Delivery Networks (CDNs) that present near-identical JA4S and JA4TS patterns. This results in fingerprint collisions and limits classifier specificity.

Embedded Webviews and SDKs: Applications using embedded browsers or third-party SDKs (e.g., advertising or analytics) often inherit TLS stack behavior from the host system or library. In such cases, distinct apps may emit indistinguishable JA4+ fingerprints.

Dynamic Behavior and Updates: Some apps exhibited fingerprint variability over time, probably due to updates. This dynamic behavior challenges the stability assumption of static fingerprinting, and makes retraining necessary every several days.

System-level Traffic Leakage: In a few cases, traffic attributed to an app was actually system-generated (e.g., update checks or push notification services), which passed through the same TLS stack but originated from different components. This introduces noise into app-specific fingerprint sets.

Rare or Sparse Apps: Applications with low traffic volume or minimal TLS interaction (e.g., utility tools or offline-first apps) produce fewer samples, limiting the classifier

ability to learn or match them reliably. These tend to be misclassified or flagged as unknown.

These case studies illustrate the boundaries of fingerprint-based classification and motivate future enhancements such as session-level context modeling, hybrid signal integration, and adaptive fingerprint tracking.

7 Discussion

The experimental results show that JA4+ fingerprints are a viable tool for mobile application identification in encrypted environments, even in the presence of modern privacy-enhancing technologies such as TLS 1.3 and ECH. This is a significant advancement in the domain of traffic classification, where reliance on cleartext metadata (e.g., SNI) is no longer feasible.

The dictionary-based classifier consistently outperforms both machine learning models, despite its conceptual simplicity. This outcome underscores that, in the context of JA4+ fingerprints, a direct mapping approach can achieve high precision with minimal overhead, making it suitable for latency-sensitive environments such as real-time traffic inspection or mobile edge computing. Its effectiveness stems not from algorithmic sophistication, but from the quality of the fingerprint design and the diversity of protocol-level signals captured. Beyond accuracy, the method offers strong operational advantages: it requires no training phase, supports incremental updates (e.g., adding fingerprints for new apps or versions), and avoids the overfitting risks typical of machine learning pipelines. Its lookup mechanism is deterministic and fast (28 µs per flow), and the memory footprint remains modest, depending only on the number and uniqueness of stored fingerprints. These properties make the classifier lightweight, maintainable, and particularly well-suited for large-scale, real-time environments such as traffic monitoring middleboxes or mobile gateways. However, the method cannot classify truly unseen or transient fingerprints unless they are explicitly recorded, which represents a trade-off between simplicity and adaptability.

While the top-1 accuracy of 58.5% may appear modest in absolute terms, it must be interpreted in light of the experimental setting. The classification task involves 50 mobile applications, all under encrypted traffic without access to SNI, and includes challenging cases with overlapping infrastructure. When considering top-3 and top-5 predictions, the accuracy improves substantially, suggesting that this method is already usable in scenarios such as traffic auditing or application grouping. Furthermore, a random classifier in this setup would only achieve 2% top-1 accuracy, highlighting the non-trivial discriminative power of the JA4+ features.

These results may also be influenced by platform-specific differences in TLS stack implementations (e.g., between Android and iOS), which can affect fingerprint variability and classification outcomes. This observation motivates future exploration of platform-tailored models and evaluation strategies.

From a privacy and ethical perspective, it is important to recognize the dual-use nature of fingerprinting techniques. While our method does not rely on decrypted content or domain names and thus improves privacy compliance compared to traditional deep packet inspection (DPI), it can still be used to infer user behavior and application us-

age patterns. In particular, large-scale deployment of such fingerprinting could enable pervasive monitoring unless safeguards are implemented. Therefore, we advocate for transparent usage policies, clearly defined objectives (e.g., threat detection, QoS enforcement), and mechanisms for oversight to ensure alignment with legal and ethical standards.

Finally, compared to the study by the University of Brno [4], our work removes the dependency on Server Name Indication (SNI), thereby aligning with a more realistic classification model under ECH adoption. Whereas their method leverages the leakage of domain names to boost accuracy, our strategy focuses entirely on behavioral and protocol-level features. This makes our approach more future-proof and privacy-compliant. Moreover, our incorporation of JA4TS and JA4X —components underexplored in prior work— results in measurable improvements in classification performance.

8 Conclusions

In this work, we presented a comprehensive study on the use of JA4+ fingerprints for encrypted mobile app identification. Our experiments, based on real-device data and extensive cross-validation, show that JA4+ enables accurate and efficient classification, even when domain name-based identifiers are unavailable.

The dictionary-based model emerges as a robust and scalable solution for this task. Nonetheless, the results also motivate future exploration of hybrid approaches that combine the precision of heuristic methods with the adaptability of learning-based models.

Future work should explore the temporal stability of JA4+ fingerprints across regions and over time, develop sequential models that incorporate fingerprint flows rather than isolated instances, and refine classifiers through platform-specific tuning. Additionally, investigating the robustness of JA4+ against evasion techniques and evaluating the scalability of dictionary-based methods for real-world deployments are key next steps. Moreover, integrating JA4+ with other traffic signals, such as DNS queries [5,10] or IP addresses [9], could provide more comprehensive classification frameworks suited for evolving privacy-enhanced protocols.

Further research could also examine strategies to reduce fingerprint collisions, particularly in ecosystems where apps share backend infrastructure or third-party SDKs. Hybrid classification techniques combining dictionary lookups with probabilistic or embedding-based models may improve accuracy and generalization. In addition, while our approach supports incremental extension, it cannot detect entirely novel traffic patterns unless explicitly recorded. Comparing JA4+ with other SNI-independent methods —such as traffic statistics like packet sizes or interarrival times— would help better position this technique within the broader landscape of encrypted traffic classification.

Dataset and code availability. The dataset and code are available in GitHub at https://github.com/martafdezterrasa/TFG_MartaFdez.

Acknowledgments. This work was supported in part by the R&D activity program with reference TEC-2024/COM-504 and acronym RAMONES-CM, granted by the Comunidad de Madrid through the Directorate General for Research and Technological Innovation via Order 5696/2024.

References

1. Aceto, G., Ciuonzo, D., Montieri, A., Pescapè, A.: MIMETIC: Mobile encrypted traffic classification using multimodal deep learning. Computer Networks **165**, 106944 (2019). https://doi.org/10.1016/j.comnet.2019.106944, accessed: 2025-05-28
2. Althouse, J.: TLS fingerprinting with JA3 and JA3S. https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967/ (2017), accessed: 2025-05-28
3. Anderson, B., McGrew, D.: Accurate TLS fingerprinting using destination context and knowledge bases. arXiv preprint arXiv:2009.01939 (2020), https://arxiv.org/abs/2009.01939, accessed: 2025-05-28
4. Burgetová, I., Ryšavý, O., Matoušek, P.: Towards identification of network applications in encrypted traffic. In: 2024 8th Cyber Security in Networking Conference (CSNet). pp. 213–221 (2024). https://doi.org/10.1109/CSNet64211.2024.10851738, accessed: 2025-05-28
5. Campo, C., Garcia-Rubio, C., Jimenez-Berenguel, A., Moure-Garrido, M., Almenares, F., Díaz-Sanchez, D.: Inferring mobile applications usage from DNS traffic. Ad Hoc Networks **163**, 103601 (2024). https://doi.org/https://doi.org/10.1016/j.adhoc.2024.103601, https://www.sciencedirect.com/science/article/pii/S1570870524002129
6. FoxIO Security: JA4+ network fingerprinting. https://github.com/FoxIO-LLC/ja4 (2023), accessed: 2025-05-28
7. Lowe, P.: Peter Lowe's ad and tracking server list. https://pgl.yoyo.org/adservers/ (2025), accessed: 2025-05-28
8. OISD Team: OISD blocklist. https://oisd.nl/ (2025), accessed: 2025-05-28
9. Perdices, D., López de Vergara, J.E., González, I., de Pedro, L.: Web browsing privacy in the deep learning era: Beyond VPNs and encryption. Computer Networks **220**, 109471 (2023). https://doi.org/https://doi.org/10.1016/j.comnet.2022.109471
10. Perdices, D., Ramos, J., García-Dorado, J.L., González, I., López de Vergara, J.E.: Natural language processing for web browsing analytics: Challenges, lessons learned, and opportunities. Computer Networks **198**, 108357 (2021). https://doi.org/https://doi.org/10.1016/j.comnet.2021.108357
11. Rescorla, E.: The transport layer security (TLS) protocol version 1.3. RFC 8446 (2018), https://www.rfc-editor.org/rfc/rfc8446, accessed: 2025-05-28
12. Rescorla, E., Oku, K., Sullivan, N., Wood, C.A.: TLS Encrypted Client Hello. Internet-Draft draft-ietf-tls-esni-24, Internet Engineering Task Force (Mar 2025), https://datatracker.ietf.org/doc/draft-ietf-tls-esni/24/, work in Progress, Accessed: 2025-05-28
13. Wireshark Foundation: Wireshark network protocol analyzer. https://www.wireshark.org (2024), version 4.2.0