NotiCorr: Exposing Social Relationships via Notification Traffic of Instant Messaging Applications

Jiangchao Chen[1,2], Zhuojun Jiang[1,2](✉), Jiangyi Yin[1,2], Dongfang Hao[1,2], Zhao Li[1,2], Meijie Du[1,2], and Qingyun Liu[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{chenjiangchao, jiangzhuojun, yinjiangyi, haodongfang, lizhao, dumeijie, liuqingyun}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

Abstract. Instant Messaging (IM) applications, such as Telegram and WeChat, have become indispensable tools for individuals. To protect users' privacy, popular IM applications employ advanced encryption mechanisms. However, we demonstrate that the encrypted traffic of popular IM applications can still leak information about users' social relationships. In this paper, we reveal that the message notification traffic in IM application is exploitable and propose a novel privacy attack called NotiCorr, which allows an adversary to infer the users in the same group based on flow correlation. Specifically, even if the IM application is not running, the client will still instantly receive group message notifications. To this end, we extract robust fingerprints from both message notification and message transmission traffic to enable attacks in more realistic usage scenarios. To the best of our knowledge, this is the first study to highlight the privacy risks posed by message notification traffic in IM applications. Through extensive experiments, we demonstrate that NotiCorr significantly outperforms related methods. Finally, we discuss the mitigation strategies.

Keywords: Social relationships · Privacy · Instant messaging applications · Flow correlation.

1 Introduction

Instant Messaging (IM) applications have become essential tools for daily communication, with over 3 billion people use mobile IM applications worldwide [15]. Due to various factors, IM applications are subject to monitoring by governments and corporations. To protect users' privacy, popular IM applications have implemented encryption technologies to secure user communications.

Despite the use of encryption mechanisms, there remain risks of privacy leakage. Some studies have employed flow correlation techniques to reveal users' privacy [5, 13, 12, 2, 3]. For example, [5, 13, 12] utilized flow correlation to perform

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97629-2_6

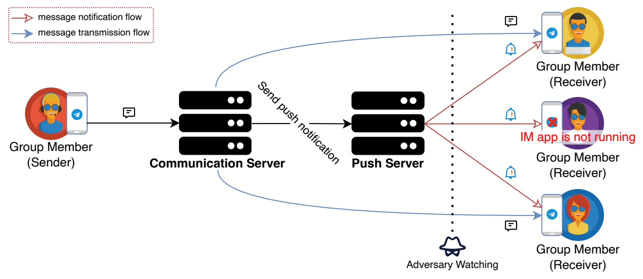


Fig. 1: The adversary correlates message notification flows and message transmission flows to identify users in the same group.

deanonymization on the Tor network. However, these methods fail to extract effective fingerprints for associating group members in IM applications. Bahramali et al. [2] and Bozorgi et al. [3] analyse the MTU-sized packets of message transmission traffic to extract fingerprints from IM application. Then, they identify users within the same group through flow correlation. However, these attacks have some limitations. When users receive messages from multiple groups simultaneously or within a short period, the correlation of message transmission flows become ineffective as the overlapping traffic disrupts the fingerprints. Additionally, these attacks are only effective when the IM application is active, as no message transmission traffic occurs when the application is not running.

To address these limitations, we find that the Push Notification Service in IM applications enables apps to deliver messages to users without requiring the app to be active. Therefore, users can receive message notifications from groups instantly even when the IM application is not running, as shown in Figure 1. Besides, the interaction patterns of the message notification flows from users within the same group exhibit notable similarities, and the fingerprints extracted from these flows are highly robust. Leveraging these insights, we propose a flow correlation based privacy attack called $NotiCorr$, which combines message notification flows and message transmission flows to identify users in the same group.

To the best of our knowledge, we are the first to leverage message notification flows and message transmission flows to carry out a privacy attack. **Initially**, NotiCorr pre-classifies each user's flows into potential message notification and transmission flows based on the packet size distribution characteristics. **Subsequently**, we propose a PING-PONG based fingerprint extraction algorithm to obtain the message notification sequence and a Packet Inter-Arrival-Time (IAT) based fingerprint extraction algorithm for the message transmission sequence. **Then**, we apply a Spatial-Temporal based Longest Common Subsequence (LCS) algorithm to extract the common message notification sequence (denoted as $cN\text{-}FP$) between clients. Additionally, we utilize merge-Dynamic Time Warping (merge-DTW) algorithm to extract the common message transmission sequence (denoted as $cD\text{-}FP$) between clients. **Ultimately**, NotiCorr extracts features from the *cN-FP* and *cD-FP* between clients and inputs these features into a well-trained classifier model to obtain the final correlation result.

Extensive experiments are conducted to evaluate our method's performance. Comparisons with related flow correlation attacks demonstrate that NotiCorr achieves a higher True Positive Rate (TPR) and a lower False Positive Rate (FPR), maintaining robust performance across varying positive and negative sample ratios. Additionally, NotiCorr is deployed on an enterprise gateway and the experimental results demonstrate its real-world practicality. Ultimately, we propose mitigation strategies to address that privacy risk.

In summary, the contributions of this paper are as follows:

– This paper is the first study to reveal that the message notification traffic flow of IM application can be used to infer users' social relationships.
– We propose a flow correlation based privacy attack called NotiCorr to correlate users in the same group. By leveraging the immediacy of message notifications, our attack remains effective even when the IM application is not running.
– We perform extensive experiments to demonstrate that NotiCorr[3] outperforms related methods. We also deploy our method at an enterprise gateway, and real-world results show its practicality, achieving a TPR greater than 0.9 and a FPR lower than 3.5×10^{-5}.

Ethics. Our analysis focuses solely on traffic from our own IM application clients, without capturing or analyzing data from others. All attacks are conducted exclusively on our own clients. In real-world experiments, we neither save raw network traffic nor analyze packet payloads. Instead, our method processes gateway traffic in real time to extract features, which are not stored afterward. Thus, our experiments do not compromise the privacy of real-world IM application clients.

2 Background & Related Work

This section first introduces the Push Notification Service, followed by a summary of current research on privacy attacks leveraging encrypted traffic analysis.

2.1 Push Notification Service

The Push Notification Service (PNS) is a widely adopted mechanism in modern IM applications, enabling the timely delivery of messages and updates to users. It comprises a cloud of push-based messaging servers that are responsible for relaying messages from application servers to clients [24]. This allows servers to send notifications directly to users' devices, ensuring that messages and updates are received instantly, even when the application is not launched.

[3] We release its source code and the datasets at https://anonymous.4open.science/r/NotiCorr-784A

2.2 Privacy Attacks Through Network Traffic Analysis

Numerous technologies have been proposed to protect user privacy, such as SSL/TLS protocols, SSH, and anonymous communication systems like Tor. Despite the increasing use of encryption, attackers can still exploit encrypted network traffic to conduct privacy attacks. Some researchers [17, 11, 13] have focused on website fingerprinting attacks, which can infer the websites a user visits. Other studies [9, 21, 7] demonstrate that attackers can identify the applications a user is using based on encrypted traffic. Moreover, some researchers [8, 14, 6] have advanced further, being able to recognize specific user activities within applications. In addition, progress has been made in content identification [19, 18, 1, 10], enabling the detection of videos users are watching or the specific webpages they are accessing.

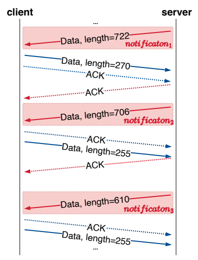
Recently, researchers have attempted to conduct privacy attacks on users' social relationships based on analysis of the encrypted traffic of IM application. Bahramali et al. [2] and Bozorgi et al. [3] utilize burst patterns in message transmission flow of IM application to extract the fingerprints, identifying users in the same group through flow correlation. These attacks are ineffective in real-world usage scenarios, such as when users receive messages from multiple group or when the application is not running. In this paper, we extract the fingerprints from the message notification flow of the IM application, which can leak more information. By combining these fingerprints with message transmission fingerprints, we implement a practical and effective privacy attack.

3 Analysing IM application message notification & message transmission traffic

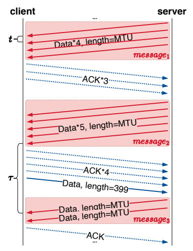
In this section, we analyse the message notification traffic and the corresponding message transmission traffic in IM applications. Among various IM applications, we select Telegram for our traffic analysis due to two primary reasons. Firstly, Telegram has over 700 million monthly active users [16], making the analysis of privacy risks in its traffic broadly impactful. Secondly, the Telegram API provides detailed information on group messages, including sending times, message types, and message sizes. Furthermore, programmable bots in Telegram allow us to customize message content, facilitating the analysis.

Initially, we create a group and add two Android clients, running Telegram in the background with message notification enabled. And a bot sends messages of varying sizes, types, and sending intervals within the group. We utilize tcpdump to capture the traffic and export notification logs from phones via Android Debug Bridge (ADB). Ultimately, we collect message notification and transmission traffic for 500 messages. After that, we use the Telegram API to retrieve message timestamps and analyze logs to extract notification timestamps, which help identify message transmission and notification traffic.

After analysing the traffic of IM application, we identify three key insights that guide the development of NotiCorr.



(a) message notification flow



(b) message transmission flow

Fig. 2: Interaction patterns of message notification flow and message transmission flow.

(i): *The message notification flow exhibits a distinct interaction pattern denoted as PING-PONG. Additionally, the downstream MTU-sized packets of the message transmission flow exhibit a distinct pattern denoted as burst.*

Our analysis reveals that message notifications are transmitted in a flow[4] with a distinct interaction pattern, as illustrated in Figure 2a. Specifically, when a message notification is generated, the server encapsulates its content in a packet payload and transmits it to the client (PING). Upon receiving the packet, the client responds with an acknowledgment packet (PONG). We refer to this packet as a message notification packet and this interaction pattern as PING-PONG. For the message transmission flows, we find that they exhibit burst patterns, as shown in Figure 2b. Specifically, the server transmits a message to the client by sending multiple MTU-sized packets within a short period. This burst pattern is consistent with findings from studies of Bahramali et al. [2] and Bozorgi et al [3]. Within each burst, packets have short inter-arrival times (e.g., $t < 0.5$ *seconds*), whereas the intervals between bursts are longer (e.g., $\tau \geq 0.5$ *seconds*).

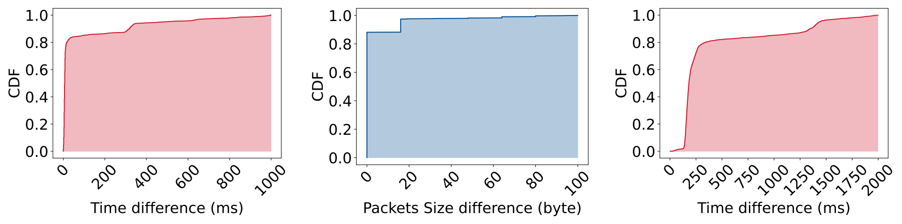
(ii): *Message notification packets from different users in the same group display highly consistent temporal (timing) and spatial (size) characteristics.*

We calculate the time and size differences of the message notification packets containing the same content between the two clients, plotting the cumulative distribution functions (CDF) in Figure 3a and Figure 3b. Figure 3a illustrates that nearly 100% of message notifications are received with a time difference of less than 1000 milliseconds, while over 80% arriving within 200 milliseconds. Figure 3b shows that almost 100% of message notification packets have a size difference of less than 100 bytes, with approximately 85% having identical sizes.

(iii): *There is a strong temporal correlation between a client's message notification packets and their corresponding message transmission packets. Exploiting this relationship enables the extraction of more robust fingerprints.*

We analyse the time distribution between a message notification packet and the corresponding message transmission on a single client. We calculate the time

[4] We define a flow as a sequence of packets composed of identical five-tuple(source/destination IP, source/destination port, transport layer protocol).



(a) (b) (c)

Fig. 3: Characterization of message notification and message transmission from two clients in the same group. 3a shows the CDF of arrival intervals of message notification packets between two clients; 3b presents the CDF of size differences of message notification packets between two clients; 3c illustrates the CDF of time intervals between message notifications and their corresponding message transmissions.

difference between the message notification packet and the corresponding message transmission, and plot the CDF, as shown in Figure 3c. Figure 3c demonstrates that approximately 80% of message notifications and their corresponding message transmissions occur within 500 milliseconds, and nearly 100% occur within 2000 milliseconds.

We also analyse other popular IM applications, like Wechat, Whatsapp and QQ, whose message notification and transmission flows exhibit similar characteristics as described above.

4 Threat Model

We assume that all traffic between IM application clients and servers is encrypted and that the message notification function within IM application is enabled. Adversaries need to monitor users' encrypted network traffic. Leveraging NotiCorr, attackers can achieve their goals without collaborating with IM providers or exploiting vulnerabilities within the application.

The attacker aims to identify the IP addresses of IM application users within the same group. By using NotiCorr, the attacker can detect IP pairs belonging to the same group and construct a social graph, revealing the social relationships of users associated with these IPs. Associating IP addresses with specific users is beyond the scope of this study and has been addressed in prior work [4].

5 Approach

In this section, we describe the methodology of this paper, referred to as NotiCorr. As shown in Figure 4, NotiCorr consists of four primary steps: (i) Preprocessing network traffic and pre-categorizing flows. (ii) Fingerprint extraction.

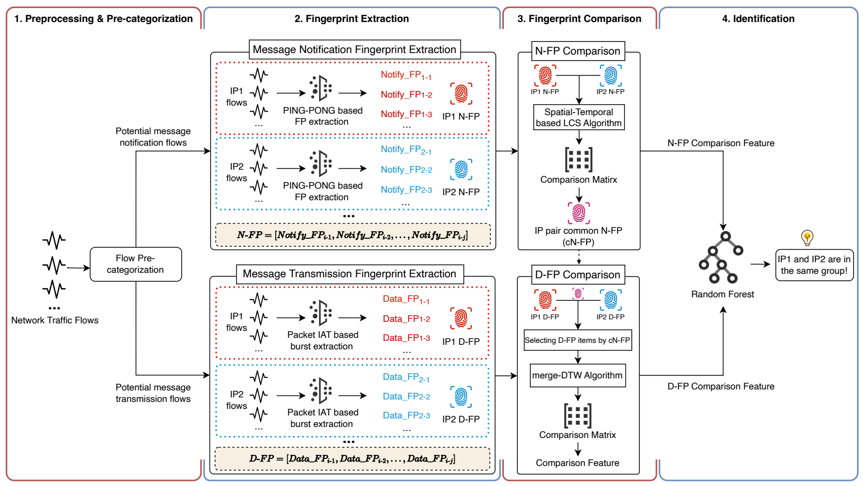

1. Preprocessing & Pre-categorization
2. Fingerprint Extraction
3. Fingerprint Comparison
4. Identification
Network Traffic Flows
Flow Pre-categorization
Potential message notification flows
Potential message transmission flows
Message Notification Fingerprint Extraction
IP1 flows
PING-PONG based FP extraction
Notify_FP1-1
Notify_FP1-2
Notify_FP1-3
IP1 N-FP
IP2 flows
Notify_FP2-1
Notify_FP2-2
Notify_FP2-3
IP2 N-FP
N-FP = [Notify_FP1-1, Notify_FP1-2, ..., Notify_FPi-j]
Message Transmission Fingerprint Extraction
Packet IAT based burst extraction
Data_FP1-1
Data_FP1-2
Data_FP1-3
IP1 D-FP
Data_FP2-1
Data_FP2-2
Data_FP2-3
IP2 D-FP
D-FP = [Data_FP1-1, Data_FP1-2, ..., Data_FPi-j]
N-FP Comparison
IP1 N-FP
IP2 N-FP
Spatial-Temporal based LCS Algorithm
Comparison Matirx
IP pair common N-FP (cN-FP)
D-FP Comparison
IP1 D-FP
IP2 D-FP
Selecting D-FP items by cN-FP
merge-DTW Algorithm
Comparison Matrix
Comparison Feature
N-FP Comparison Feature
D-FP Comparison Feature
Random Forest
IP1 and IP2 are in the same group!

Fig. 4: The framework of NotiCorr.

(iii) Fingerprint comparison. Then, we extract the comparison features from these sequences. (iv) Identification.

5.1 Network Preprocessing and Flow Pre-categorization

In this step, network traffic is initially divided into flows. We then classify these flows into either potential message notification flows or potential message transmission flows based on the proportion of MTU-sized packets in the downstream of each flow. A detailed analysis of the packet size distribution for message notification flows and other flows is provided in Section 6.

5.2 Fingerprint Extraction

Based on the analysis in Section 3, we develop distinct fingerprint extraction algorithms for message notification and transmission flows.

PING-PONG based fingerprint extraction. For a given flow j of a client IP i, the objective of this algorithm is to extract the message notification sequence as a fingerprint $Notify\_FP_{i\text{-}j} = [(NT_{i\text{-}j_1}, NS_{i\text{-}j_1}), (NT_{i\text{-}j_2}, NS_{i\text{-}j_2}), ..., (NT_{i\text{-}j_n}, NS_{i\text{-}j_n})]$, where $NT_{i\text{-}j_n}$ represents the timestamp of the n-th message notification and $NS_{i\text{-}j_n}$ denotes its size. As detailed in Section 3, each message notification packet sent from the server to the client is typically followed by an ACK packet from the client to the server. Consequently, we identify message notification packets by locating downstream packets that arrive between two consecutive upstream packets in the potential message notification flow. We denote the downstream packet time as the message notification time and packet size as the message notification size. If there are multiple downstream packets between two consecutive upstream packets, the message notification size is determined by summing the sizes of these downstream packets, and the message notification time is marked by the arrival time of the first downstream packet. This method allows an adversary to extract potential message notification sequences from the observed message notification flows associated with a given client IP.

Packets IAT based fingerprint extraction. When an IM client receives a message (e.g., text, picture, video, audio, etc.), the downstream packets of message transmission flow exhibit burst pattern. This feature has also been analyzed in previous studies [3, 2]. In this paper, the burst sequence extracted from the message transmission flow as message transmission fingerprint is an auxiliary feature to reduce the false positive rate. We extract burst sequence from the MTU-sized packets of message transmission downstream based on packets IAT. After extracting fingerprint based on packets IAT, we get the burst sequence $Data\_FP_{i\text{-}j} = [(DT_{i\text{-}j_1}, DS_{i\text{-}j_1}), (DT_{i\text{-}j_2}, DS_{i\text{-}j_2}), ..., (DT_{i\text{-}j_n}, DS_{i\text{-}j_n})]$ from message transmission flow j of client IP i, where $DT_{i\text{-}j_n}$ represents the timestamp of the first packet in the n-th burst and $DS_{i\text{-}j_n}$ denotes the cumulative packet size in that burst.

After fingerprint extraction, each flow of one client is extracted either a notification fingerprint or a message transmission fingerprint. Ultimately, we construct a list of notification fingerprints (denoted as $N\text{-}FP$) and a list of message transmission fingerprints (denoted as $D\text{-}FP$) for all flows of each client.

5.3 Fingerprint Comparison

This step aims to derive message notification comparison features denoted as $\mathcal{F}_{N\text{-}FP}$, and message transmission comparison features denoted as $\mathcal{F}_{D\text{-}FP}$, by comparing the message notification fingerprint lists and message transmission fingerprint lists between each client IP pair.

Message Notification Fingerprint Comparison. According to Section 3, message notification packets of clients from the same group exhibit temporal and spatial similarities. Therefore, given two message notification fingerprint items $(NT_{p\text{-}j_m}, NS_{p\text{-}j_m})$ and $(NT_{q\text{-}k_n}, NS_{q\text{-}k_n})$, we consider these items to be similar if $|NT_{p\text{-}j_m} - NT_{q\text{-}k_n}| < \alpha$ and $|NS_{p\text{-}j_m} - NS_{q\text{-}k_n}| < \beta$, where α and β are the thresholds of time and size. Based on this logic, we design a Spatial-Temporal based LCS algorithm to identify the common message notification sequence between IP pair, as shown in our source code. By comparing the message notification fingerprints between a client IP pair using this algorithm, we finally obtain a comparison matrix and we denote the element in the k-th row and j-th column of this matrix as $cN\text{-}FP_{k\text{-}j} = (cN\text{-}FP_{k\text{-}j_p}, cN\text{-}FP_{k\text{-}j_q})$, where $cN\text{-}FP_{k\text{-}j_p}$ is the common message notification sequence of flow j from client IP p and $cN\text{-}FP_{k\text{-}j_q}$ is the common message notification sequence of flow k from client IP q. We sort all message notification sequence items of each IP in the comparison matrix by time to derive the common message notification sequence denoted as $cN\text{-}FP$.

Subsequently, we calculate statistical features from $cN\text{-}FP$ to form the $\mathcal{F}_{N\text{-}FP}$. These features include the length of $cN\text{-}FP$ and the mean, minimum, and maximum values of $mean(cN\text{-}FP)$. The $mean(cN\text{-}FP)$ is defined as follows:

$$mean(cN\text{-}FP) = [\frac{S_{p1} + S_{q1}}{2}, \frac{S_{p2} + S_{q2}}{2}, ..., \frac{S_{pn} + S_{qn}}{2}]$$

where S_{pn} and S_{qn} are the sizes of the n-th item in $cN\text{-}FP$ for client IPs p and q, respectively.

Message Transmission Fingerprint Comparison. Figure 3c demonstrates that message notification packets and corresponding message transmissions exhibit a strong temporal correlation. Consequently, we select the burst from message transmission fingerprint based on the time of the item in $cN\text{-}FP$. Specifically, given a message transmission fingerprint item $(DT_{p\text{-}j_m}, DS_{p\text{-}j_m})$, and an item (T_{pn}, S_{pn}) in $cN\text{-}FP$ of client IP p, if $T_{pn} - \tau_{-} < DT_{p\text{-}j_m} < T_{pn} - \tau_{+}$, we consider that there is a temporal correlation between $(DT_{p\text{-}j_m}, DS_{p\text{-}j_m})$ and (T_{pn}, S_{pn}), where T_{pn} is time of the n-th item in $cN\text{-}FP$, τ_{-} and τ_{+} are predetermined thresholds. After that, we obtain the selected message transmission fingerprint list $\widetilde{D\text{-}FP}_p$ and $\widetilde{D\text{-}FP}_q$ for a client IP pair p and q, respectively. We then apply our merge-DTW algorithm to identify the most similar burst sequence and calculate the corresponding distance between $\widetilde{D\text{-}FP}_p$ and $\widetilde{D\text{-}FP}_q$. By comparing the message transmission fingerprints between a client IP pair, we construct a comparison matrix and denote the element in the k-th row and j-th column of this matrix as $cD\text{-}FP_{k\text{-}j} = (dis_{k\text{-}j}, cN\text{-}FP_{k\text{-}j_p}, cN\text{-}FP_{k\text{-}j_q})$, where $cN\text{-}FP_{k\text{-}j_p}$ and $cN\text{-}FP_{k\text{-}j_q}$ are the common burst sequences of the client IP pair p and q, $dis_{k\text{-}j}$ is the distance between $cN\text{-}FP_{k\text{-}j_p}$ and $cN\text{-}FP_{k\text{-}j_q}$. Subsequently, we identify a matrix element with the minimum distance dis denoted as $min(cD\text{-}FP)$. We then calculate the total length, maximum value, and mean value of the two common burst sequences in $min(cD\text{-}FP)$. These features serve as the $\mathcal{F}_{D\text{-}FP}$.

Merge-DTW is an optimized version of the standard DTW algorithm that we developed to account for the effects of network conditions such as network latency, which may cause a single message to be split into two bursts in flow. Specifically, when calculating the distance between two burst items $(DT_{p\text{-}j_m}, DS_{p\text{-}j_m})$ and $(DT_{q\text{-}k_n}, DS_{q\text{-}k_n})$, merge-DTW computes the absolute difference between $DS_{p\text{-}j_m}$ and $DS_{q\text{-}k_n}$. Additionally, it calculates the absolute difference between $DS_{p\text{-}j_m}$ and the sum of $DS_{q\text{-}k_n}$ and $DS_{q\text{-}k_{n\text{-}1}}$. The minimum value of these two absolute difference values is then taken as the distance between two burst items.

5.4 Identification

In this step, we have derived the comparison features $\mathcal{F}_{D\text{-}FP}$ and $\mathcal{F}_{N\text{-}FP}$ for a given client IP pair. These features are used as inputs to a classifier, which outputs the probability that the IP pair belongs to the same group. We finally select the Random Forest model as the classifier.

Table 1: Group relationships of Telegram clients.

Clients	$group_A$	$group_B$	$group_C$	$group_D$	$group_E$
C_1	yes	yes	no	no	no
C_2	no	yes	yes	yes	no
C_3	no	no	yes	no	yes

6 Evaluation

In this section, we evaluate the performance of NotiCorr under various configurations and compare its effectiveness against related methods.

6.1 Experimental Setup

Dataset. We simulate group relationships using three Telegram clients, where two client pairs share the same groups, and each client also participates in other groups independently, as shown in Table 1. Each client remains active with message notifications enabled. And Telegram bots send messages in these groups, generating the traffic of message notification and message transmission. To simulate realistic group chat scenarios, we add 500 public groups and record message features, including message time, type and size, using the Telegram API. Following the method of Bahramali et al. [2], we generate message sequences for each group, including sending intervals, message types and sizes. In each round, bots send these messages and we capture traffic from each client using tcpdump until all messages are sent, resulting in 8,000 rounds of traffic with 126,798 flows.

Since our evaluation requires ground truth labels, we export Telegram's message notification logs via ADB. We label the message notification flows in each round of traffic using the log information. We divide the traffic corresponding to 8000 rounds into training, testing, and validation datasets using a 7:1:2 ratio.

Baseline. The most relevant methods to NotiCorr are traffic flow correlation. Therefore, we compare NotiCorr with existing privacy attack methods that utilize flow correlation, which either have open source code or provide the source code through contact with the authors. These related methods include DeepCorr [12], DeepCoffea [13], FlowTracker [5] and Bahramali [2].

Metrics. Similar to previous studies [2, 12, 13], True Positive Rate (TPR) and False Positive Rate (FPR) are used to evaluate flow correlation performance Due to the TPR and FPR of NotiCorr, DeepCorr, DeepCoffea and Bahramli vary with changes in the threshold. Therefore, we also calculate the area under the curve (AUC) based on their TPR and FPR at different thresholds to measure their overall performance. The closer a method's AUC value is to 1, the higher its TPR and the lower its FPR at a given threshold.

6.2 NotiCorr Effectiveness

We evaluate the impact of the different settings and ideas behind NotiCorr on the overall effectiveness of the attack.

Table 2: Hyperparameters of NotiCorr. (Note: The bold numbers in the search range represent the default values of the parameters)

threshold	search range	optimal value
$ratio_{D-MTU}$	✘	0.4
IAT_{burst}	✘	0.5 seconds
α	[1000, **2000**, ..., 5000]	1000
β	[0, 100, **200**, ..., 400]	0
τ_-	[1000, **2000**, ..., 5000]	2000
τ_+	[1000, **2000**, ..., 5000]	2000

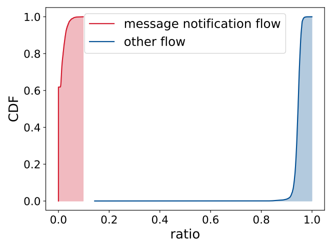


Fig. 5: CDF graph of the proportion of MTU size packets in the downstream of message notification flows and other flows

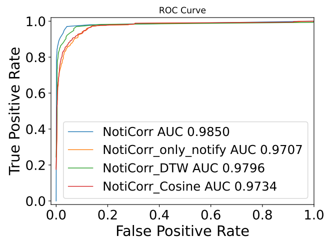


Fig. 6: Comparison of the effectiveness of using different algorithms for D-FP comparison in NotiCorr.

Hyperparameter Selection. There are multiple thresholds in NotiCorr, including the ratio of the number of MTU-size packets in downstream flow for *Flow Pre-categorization* (denoted as $ratio_{D\text{-}MTU}$), the IAT of packets in Section 5.2 (denoted as IAT_{burst}), the time threshold α and size threshold β in Section 5.3, the time range threshold τ_- and τ_+ in Section 5.3. For the threshold $ratio_{D\text{-}MTU}$, we conduct a statistical analysis on the proportion of MTU-sized packets in the downstream of message notification flows and the other flows in the training and testing datasets, as shown in Figure 5. The proportion of MTU-sized packets shows distinct difference between message notification flows and other flows. Therefore, we empirically select 0.4 as the value of threshold $ratio_{D\text{-}MTU}$. For threshold IAT_{burst}, we use the value of 0.5 seconds from the work of Bahramali et al. [2] And for the remaining thresholds α, β, τ_- and τ_+, we search within a certain range for each threshold in the training and testing datasets to find the value that maximize the effectiveness of NotiCorr. Specifically, we iterate through each parameter, change its value in the search space each time, and train a random forest model in NotiCorr to obtain its best performance in testing dataset. Table 2 shows the search range and optimal value of each parameter in NotiCorr.

Table 3: The performance of NotiCorr and other four methods with different positive and negative sample ratios. The values in brackets are the performance of methods using our features

method	metrics	positive and negative sample ratio				
		1:9	1:49	1:99	1.149	1.199
NotiCorr	AUC	**0.9822**	**0.9820**	**0.9820**	**0.9822**	**0.9822**
DeepCorr	AUC	0.7321(0.9700)	0.6758(0.9668)	0.6363(0.9695)	0.5702(0.9690)	0.4894(0.9667)
DeepCoffea	AUC	0.6227(0.9507)	0.5979(0.9506)	0.5789(0.9504)	0.5327(0.9492)	0.5259(0.9438)
Bahramali	AUC	0.7685(0.9659)	0.6945(0.9599)	0.6689(0.9546)	0.5698(0.9539)	0.5064(0.9467)
FlowTracker	TPR	0.5083(0.9698)	0.4368(0.9584)	0.4232(0.9509)	0.4209(0.9498)	0.4168(0.9550)
	FPR	0.2991(0.0764)	0.2197(0.0662)	0.2099(0.0640)	0.2066(0.0635)	0.2050(0.0635)

The effectiveness of message transmission fingerprint and message notification fingerprint. To evaluate the effectiveness of the two fingerprint types in NotiCorr, we compare its performance with a version using only the message notification fingerprint, denoted as NotiCorr_only_notify. As shown in Figure 6, NotiCorr_only_notify achieves an AUC of 0.9707, demonstrating the effectiveness of the message notification fingerprint. When the TPR is 0.9, the FPR of NotiCorr_only_notify is higher than that of NotiCorr, indicating that message transmission fingerprint can reduce the false positive rate.

The effectiveness of merge-DTW. To evaluate the effectiveness of merge-DTW, we compare the performance of NotiCorr, NotiCorr using standard DTW algorithm (denoted as NotiCorr_DTW), and NotiCorr using cosine similarity (denoted as NotiCorr_Cosine). The results, shown in Figure 6, indicate that NotiCorr using merge-DTW outperforms NotiCorr_DTW. This improvement is due to merge-DTW accounting for scenarios where a message transmission burst may split into two bursts due to network environment, such as network latency. Additionally, NotiCorr using merge-DTW outperforms NotiCorr_Cosine, as merge-DTW allows for non-linear alignment of two sequences, enabling the optimal matching for sequences with unequal lengths or time axis offsets.

6.3 Comparison with related methods

In this section, we compare NotiCorr with four related methods on our dataset. Table 3 shows the performance of NotiCorr and four other methods with different positive and negative sample ratios. From the table, we can draw the following conclusions:

NotiCorr achieves a greater TPR with a lower FPR. Across all positive and negative sample ratios, NotiCorr achieves a greater TPR with a lower FPR compared to the four other methods. The success of NotiCorr in achieving such strong experimental results can be attributed to its effective utilization of the PING-PONG interaction pattern in message notification traffic and the burst characteristics of message transmission traffic during the fingerprint extraction phase, enabling the accurate identification of both message notification and message transmission packets.

To further verify the effectiveness of the feature extracted in this paper, we use the common message notification sequence $cN\text{-}FP$ of each IP pair as the input to baseline methods. We then train and evaluate these four methods, and the results are shown in Table 3. According to the result, our features significantly improve the performance of all four methods across different sample ratios. This indicates that the original features used in these methods are not well-suited for the attack scenario described in this paper.

NotiCorr exhibits robustness under different positive and negative sample ratios. As shown in Table 3, even as the proportion of negative samples increases, the AUC for NotiCorr remains consistently high (AUC ≥ 0.98). In contrast, the AUC of the three other methods decreases as the proportion of negative samples rises, and the effectiveness of FlowTracker decreases with the increase in the proportion of negative samples.

6.4 Real-world Experiments

In this section, we evaluate the effectiveness of our method in a real-world network environment. Specifically, we collaborate with an enterprise to deploy NotiCorr at the enterprise gateway and evaluate its effectiveness in a practical setting.

We monitor the traffic continuously, segmenting it into 30-second time windows. Within each window, flows are preprocessed and pre-categorized, and fingerprints are extracted. Next, NotiCorr's Fingerprint Comparison and Identification modules were employed to correlate potential notification flow pairs. Upon correlation, NotiCorr provided the client IPs of the flow pair. Twenty enterprise volunteers are recruited and permitted to freely form groups and join public Telegram groups, with their usage information recorded.

Our method is deployed continuously at the enterprise gateway for 7 days, during which we record the total time taken for correlation within a window. **NotiCorr takes less than 15 seconds to process each window, ensuring that the current window could be processed before the next one is generated**. NotiCorr's output for each window is then compared with volunteer usage data to evaluate its real-world performance. The results show that NotiCorr achieves a TPR above 0.9 with a FPR below 3.5×10^{-5}.

7 Mitigation strategies

We propose two types of mitigation strategies for IM application users and providers. For IM application providers, we recommend altering the interaction pattern of the message notification traffic. One approach is to introduce redundant packets into the notification flows, thereby making notification packets indistinguishable. For IM application users, we suggest using proxy or VPN tools (such as V2Ray/V2Fly and shadowTLS) with obfuscation configurations that include multiplexing and random padding. Multiplexing interleaves packets from multiple flows, altering traffic patterns in terms of packet size, timing, and direction [22]. By mixing multiple client flows, multiplexing makes it challenging for attackers to distinguish individual flows. Additionally, random padding involves appending dummy data to payloads to obscure patterns of packet sizes [20, 23]. This technique alters the fingerprints that attackers can extract, thereby complicating successful fingerprint comparison.

8 Conclusion

In this paper, we reveal the message notification traffic in IM applications is vulnerable, which can leak users' privacy. Specifically, we design a privacy attack, called NotiCorr, based on message notification fingerprints and message transmission fingerprints, which allows attackers to identify users in the same group. Through extensive experiments, we demonstrate that the fingerprints extracted by NotiCorr are robust and our method outperforms related methods. We also deploy NotiCorr at an enterprise gateway to evaluate its performance and the results indicate that our method is practical and effective in the wild. Moreover, we propose corresponding mitigation strategies to IM application providers and users for this privacy attack.

Acknowledgments. This work is supported by the Scaling Program of Institute of Information Engineering, CAS (Grant No. E3Z0041101)

References

1. Bae, S., Son, M., Kim, D., Park, C., Lee, J., Son, S., Kim, Y.: Watching the watchers: Practical video identification attack in LTE networks. In: 31st USENIX Security Symposium, USENIX Security 2022. pp. 1307–1324 (2022)
2. Bahramali, A., Houmansadr, A., Soltani, R., Goeckel, D., Towsley, D.: Practical traffic analysis attacks on secure messaging applications. In: 27th Annual Network and Distributed System Security Symposium, NDSS 2020 (2020)
3. Bozorgi, A., Bahramali, A., Rezaei, F., Ghafari, A., Houmansadr, A., Soltani, R., Goeckel, D., Towsley, D.: I still know what you did last summer: Inferring sensitive user activities on messaging applications through traffic analysis. IEEE Trans. Dependable Secur. Comput. **20**(5), 4135–4153 (2023)
4. Cui, T., Gou, G., Xiong, G., Li, Z., Cui, M., Liu, C.: SiamHAN: IPv6 address correlation attacks on TLS encrypted traffic via siamese heterogeneous graph attention network. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 4329–4346 (2021)
5. Guan, Z., Liu, C., Xiong, G., Li, Z., Gou, G.: Flowtracker: Improved flow correlation attacks with denoising and contrastive learning. Comput. Secur. **125**, 103018 (2023)
6. Hu, X., Shu, Z., Tong, Z., Cheng, G., Li, R., Wu, H.: Fine-grained ethereum behavior identification via encrypted traffic analysis with serialized backward inference. Comput. Networks **237**, 110110 (2023)
7. Li, J., Wu, S., Zhou, H., Luo, X., Wang, T., Liu, Y., Ma, X.: Packet-level open-world app fingerprinting on wireless traffic. In: 29th Annual Network and Distributed System Security Symposium, NDSS 2022 (2022)

8. Li, J., Zhou, H., Wu, S., Luo, X., Wang, T., Zhan, X., Ma, X.: FOAP: fine-grained open-world android app fingerprinting. In: Butler, K.R.B., Thomas, K. (eds.) 31st USENIX Security Symposium, USENIX Security 2022. pp. 1579–1596 (2022)
9. Li, Z., Xu, X.: L2-bitcn-cnn: Spatio-temporal features fusion-based multi-classification model for various internet applications identification. Comput. Networks **243**, 110298 (2024)
10. Mitra, G., Vairam, P.K., Saha, S., Chandrachoodan, N., Kamakoti, V.: Snoopy: A webpage fingerprinting framework with finite query model for mass-surveillance. IEEE Trans. Dependable Secur. Comput. **20**(5), 3734–3752 (2023)
11. Mitseva, A., Panchenko, A.: Stop, don't click here anymore: Boosting website fingerprinting by considering sets of subpages. In: 33rd USENIX Security Symposium (USENIX Security 24). pp. 4139–4156. Philadelphia, PA (Aug 2024)
12. Nasr, M., Bahramali, A., Houmansadr, A.: Deepcorr: Strong flow correlation attacks on tor using deep learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018. pp. 1962–1976. ACM (2018)
13. Oh, S.E., Yang, T., Mathews, N., Holland, J.K., Rahman, M.S., Hopper, N., Wright, M.: Deepcoffea: Improved flow correlation attacks on tor via metric learning and amplification. In: 43rd IEEE Symposium on Security and Privacy, SP 2022. pp. 1915–1932. IEEE (2022)
14. Shan, Y., Cheng, G., Chen, Z.: Identifying fine-grained douyin user behaviors via analyzing encrypted network traffic. In: 19th International Conference on Mobility, Sensing and Networking, MSN 2023, December 14-16. pp. 868–875. IEEE (2023)
15. Statista: Number of mobile messaging users worldwide. https://www.statista.com/statistics/483255/number-of-mobile-messaging-users-worldwide/ (2023), accessed: 2024-08-31
16. Telegram: 700 million users and telegram premium. https://telegram.org/blog/700-million-and-premium (2022), accessed: 2024-08-31
17. Wang, T.: High precision open-world website fingerprinting. In: 2020 IEEE Symposium on Security and Privacy, SP 2020. pp. 152–167. IEEE (2020)
18. Wu, H., Li, X., Cheng, G., Hu, X.: Monitoring video resolution of adaptive encrypted video traffic based on HTTP/2 features. In: 2021 IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2021. pp. 1–6 (2021)
19. Wu, H., Yu, Z., Cheng, G., Guo, S.: Identification of encrypted video streaming based on differential fingerprints. In: 39th IEEE Conference on Computer Communications, INFOCOM Workshops 2020. pp. 74–79 (2020)
20. XTLS: Xtls vision, fixes tls in tls, to the star and beyond · xtls/xray-core · discussion 1295. https://github.com/XTLS/Xray-core/discussions/1295 (2023), accessed: 2024-08-31
21. Xu, H., Li, S., Cheng, Z., Qin, R., Xie, J., Sun, P.: Trafficgcn: Mobile application encrypted traffic classification based on GCN. In: IEEE Global Communications Conference, GLOBECOM 2022. pp. 891–896. IEEE (2022)
22. Xue, D., Kallitsis, M., Houmansadr, A., Ensafi, R.: Fingerprinting obfuscated proxy traffic with encapsulated TLS handshakes. In: 33rd USENIX Security Symposium, USENIX Security 2024 (2024)
23. Yawning: obfs4. https://gitlab.com/yawning/obfs4 (2023), accessed: 2024-08-31
24. Zhao, S., Lee, P.P.C., Lui, J.C.S., Guan, X., Ma, X., Tao, J.: Cloud-based push-styled mobile botnets: a case study of exploiting the cloud to device messaging service. In: 28th Annual Computer Security Applications Conference, ACSAC 2012. pp. 119–128 (2012)