Practical Traffic Analysis Attacks on Secure Messaging Applications

Alireza Bahramali*, Amir Houmansadr*, Ramin Soltani†, Dennis Goeckel‡, and Don Towsley*
University of Massachusetts Amherst
*{abahramali, amir, towsley}@cs.umass.edu
†ramin.soltani@gmail.com
‡dgoeckel@engin.umass.edu

***Abstract*—Instant Messaging (IM) applications like Telegram, Signal, and WhatsApp have become extremely popular in recent years. Unfortunately, such IM services have been targets of continuous governmental surveillance and censorship, as these services are home to public and private communication channels on socially and politically sensitive topics. To protect their clients, popular IM services deploy state-of-the-art encryption mechanisms. In this paper, we show that despite the use of advanced encryption, popular IM applications leak sensitive information about their clients to adversaries who merely monitor their encrypted IM traffic, with no need for leveraging any software vulnerabilities of IM applications. Specifically, we devise traffic analysis attacks that enable an adversary to identify administrators as well as members of target IM channels (e.g., forums) with high accuracies. We believe that our study demonstrates a significant, real-world threat to the users of such services given the increasing attempts by oppressive governments at cracking down controversial IM channels.**

We demonstrate the practicality of our traffic analysis attacks through extensive experiments on real-world IM communications. We show that standard countermeasure techniques such as adding cover traffic can degrade the effectiveness of the attacks we introduce in this paper. We hope that our study will encourage IM providers to integrate effective traffic obfuscation countermeasures into their software. In the meantime, we have designed and deployed an open-source, publicly available countermeasure system, called IMProxy, that can be used by IM clients with no need for any support from IM providers. We have demonstrated the effectiveness of IMProxy through experiments.

I. Introduction

Instant Messaging (IM) applications like Telegram, Signal, and WhatsApp have become enormously popular in recent years. Recent studies estimate that over 2 billion people use mobile IM applications across the world [42]. IM services enable users to form private and public social groups and exchange messages of various types, including text messages, images, videos, and audio files. In particular, IM applications are used extensively to exchange politically and socially sensitive content. As a result, governments and corporations increasingly monitor the communications made through popular IM services [1], [2], [73], [85].

Network and Distributed Systems Security (NDSS) Symposium 2020
23-26 February 2020, San Diego, CA, USA
ISBN 1-891562-61-4
https://dx.doi.org/10.14722/ndss.2020.24347
www.ndss-symposium.org

A notable example of oppressed IM services is Telegram [83] with over 200 million users globally [61], where a large fraction of its users come from countries with strict media regulations like Iran and Russia. In particular, Telegram is so popular in Iran that it has been estimated to consume more than 60 percent of Iran's Internet bandwidth [9]. Consequently, Iranian officials have taken various measures to monitor and block Telegram: from requesting Telegram to host some of its servers inside Iran to enable surveillance [85], to requesting Telegram to remove controversial political and non-political channels [85]. Eventually, Iran blocked Telegram entirely in April 2018 due to Telegram's non-compliance. Despite this, statistics suggest only a small decrease in Telegram's Iranian users who connect to it through various kinds of VPNs [43]. Telegram has also been blocked in Russia as Telegram operators refrained from handing over their encryption keys to Russian officials for surveillance [73]. Finally, in the light of Telegram's crucial role in recent Hong Kong protests, there are unconfirmed reports [22], [75] that mainland Chinese and Hong Kong authorities may have attempted to discover Hong Kong protesters by misusing a Telegram feature that enabled them to map phone numbers to Telegram IDs.

A Fundamental Vulnerability: Popular IM applications like Telegram, WhatsApp, and Signal deploy encryption (either end-to-end or end-to-middle) to secure user communications. We refer to such services as *secure IM (SIM)* applications. In this paper, we demonstrate that *despite their use of advanced encryption, popular IM applications leak sensitive information about their clients' activities to surveillance parties*. Specifically, we demonstrate that surveillance parties are capable of identifying members as well as administrators[1] of target IM communications (e.g., politically sensitive IM channels) with very high accuracies, and by only using low-cost traffic analysis techniques. Note that our attacks are *not* due to security flaws or buggy software implementations such as those discovered previously [32], [49], [74], [94]; while important, such security flaws are scarce, and are immediately fixed by IM providers once discovered. Instead, our attacks enable surveillance by *merely watching encrypted IM traffic* of IM users, and assuming that the underlying IM software is entirely secure. The key enabler of our attacks is the fact that major IM operators *do not* deploy any mechanisms to obfuscate traffic characteristics (e.g., packet timing and sizes), due to the impact of obfuscation on the usability and performance of

[1]An administrator of an IM channel is a member who is privileged to post messages to that channel.

such services. We therefore argue that *our attacks demonstrate a fundamental vulnerability in major in-the-wild IM services*, and, as we will demonstrate, they work against all major IM services.

We believe that our attacks present *significant real-world threats* to the users of believed-to-be-secure IM services, specially given escalating attempts by oppressive regimes to crack down on such services, e.g., the recent attempts [1], [2], [22], [75] to identify and seize the administrators and members of controversial IM communications.

Our Contributions: We design traffic analysis attack algorithms for SIM communications; the objective of our attack is to *identify the admins and/or members* of target SIM communications. What enables our attack is that, *widely-used SIM services do not employ any mechanisms to obfuscate statistical characteristics of their communications*.

We start by establishing a statistical model for IM traffic characteristics. Such a model is essential in our search for effective traffic analysis attacks on SIM services. To model IM communications, we join over 1,000 public Telegram channels and record their communications, based on which we derive a statistical model for IM traffic features. We use our SIM model to derive theoretical bounds on the effectiveness of our traffic analysis algorithms; we also use our statistical model to generate arbitrary numbers of synthetic SIM channels to enhance the confidence of our empirical evaluations.

Based on our statistical model for IM communications, we use hypothesis testing [68] to *systematically* design effective traffic analysis attack algorithms. Specifically, we design two traffic analysis attack algorithms; our first algorithm, which we call the *event-based* correlator, relies on the statistical model that we derive for SIM communications to offer an optimal matching of users to channels. Our second algorithm, which we call the *shape-based* algorithm, correlates the shapes of SIM traffic flows in order to match users to target channels. Our shape-based algorithm is slower but offers more accurate detection performance than the event-based algorithm. In practice, the adversary can cascade the two algorithms to optimize computation cost (and scalability) versus detection performance. Note that, as demonstrated through experiments, our statistical detectors outperform deep learning based detectors trained on IM traffic. This is because, as also demonstrated in recent work [58], deep learning traffic classifiers outperform statistical classifiers *only* in network applications with non-stationary noise conditions (e.g., Tor), where statistical models becomes unreliable.

We perform extensive experiments on live and synthetic SIM traffic to evaluate the performance of our attacks. We demonstrate that our algorithms offer extremely high accuracies in disclosing the participants of target SIM communications. In particular, we show that *only 15 minutes* of Telegram traffic suffices for our shape-based detector to identify the admin of a target SIM channel with a 94% accuracy and a 10^{-3} false positive rate—the adversary can reduce the false positive to 5×10^{-5} by observing an hour of traffic (the adversary can do this hierarchically, e.g., by monitoring the users flagged using 15 min of traffic for longer traffic intervals).

We also study the use of standard traffic analysis countermeasures against our attacks. In particular, we investigate tunneling SIM traffic through VPNs, mixing it with background traffic, adding cover IM traffic, and delaying IM packets. As expected, our experiments show that such countermeasures reduce the effectiveness of the attacks at the cost of additional communication overhead as well as increased latency for SIM communications. For instance, we find that tunneling SIM traffic through VPN and mixing it with background web-browsing traffic reduces the accuracy of our attack from 93% to 70%, and adding cover traffic with a 17% overhead drops the accuracy to 62%. We argue that since many SIM users do not deploy such third-party countermeasures due to usability reasons, SIM providers should integrate standard traffic obfuscation techniques into their software to protect their users against the introduced traffic analysis attacks. In the meantime, *we have designed and deployed an open-source, publicly available countermeasure system, called IMProxy, that can be used by IM clients with no need to any support from IM providers*. We have demonstrated the effectiveness of IMProxy through experiments.

In summary, we make the following contributions:

- We introduce traffic analysis attacks that reliably identify users involved in sensitive communications through secure IM services. To launch our attacks, the adversary does not need cooperate with IM providers, nor does he need to leverage any security flaws of the target IM services.
- We establish a statistical model for regular IM communications by analyzing IM traffic from a large number of real-world IM channels.
- We perform extensive experiments on the popular IM services of Telegram, WhatsApp, and Signal to demonstrate the in-the-wild effectiveness of our attacks.
- We study potential countermeasures against our attacks. In particular, we design and deploy IMProxy, which is an open-source, publicly available countermeasure system. IMProxy works for all major IM services, with no need to any support from IM providers.

II. Background: Secure Instant Messaging (SIM) Applications

We define a **secure IM (SIM)** service to be an instant messaging service that satisfies two properties: (1) it deploys strong encryption on its user communications (either end-to-end or end-to-middle), and (2) it is not controlled or operated by an adversary, e.g., a government. While our attacks also apply to non-secure IM applications, an adversary can use other trivial techniques to compromise privacy of non-secure IM services. For instance, if the operator of an IM service fully cooperates with a surveillant government, e.g., the *WeChat* IM service in China or the *Soroush* service in Iran, the IM provider can let the adversary identify target users with no need for traffic analysis mechanisms. Similarly, an IM service with weak encryption can be trivially eavesdropped with no need for sophisticated traffic analysis attacks.

Table I overviews some of the most popular SIM services.

A. How SIM Services Operate

Architecture: All major IM services are *centralized*, as shown in Table I. Therefore, all user communications in such services

TABLE I: Popular IM services [79]

IM Service	Monthly Users	Main Servers Hosted in	Owned by	End-to-End Encryption	Centralized
WhatsApp	1300 M	United States	Facebook	✓	✓
Facebook Messenger	1300 M	United States	Facebook	✓(Secret Communications)	✓
WeChat	980 M	China	Tencent	✗	✓
QQ Mobile	843 M	China	Tencent	✓	✓
Skype	300 M	Estonia	Microsoft	✓	✓
Viber	260 M	Luxembourg	Rakuten	✓Since 2016	✓
Snapchat	255 M	United States	Snap Inc	✗	✓
LINE	203 M	Japan	Line Corporation	✓	✓
Telegram	200 M	UAE	Telegram Messenger LLP	✓(Secret Chats)	✓
Signal	10 M	United States	Open Whisper Systems	✓	✓

are exchanged through servers hosted by the IM provider companies, e.g., Telegram Messenger LLP (note that some less popular services use a peer-to-peer architecture, e.g., FireChat [80], Ring [72], and Briar [15]). Each IM service has a server for authentication and key exchange. A database server stores message contents and other user information (possibly encrypted with client keys). Some IMs use Content Delivery Networks (CDNs) to run their databases to improve quality of service and resist attacks. Existing IM services use various messaging protocols for user communications, including Signal [31], Matrix [8], MTProto [55], and Off-the-Record [14]. Each of these protocols involves several stages including authentication, key exchange, message transmission, re-keying, and MAC key publishing [44].

Popular IM services intermediate all user communications by having user traffic go through their servers. Such a centralized architecture allows IM providers to offer high quality of services and solves critical issues like reaching to offline clients and clients behind NAT/firewalls. However, this presents different privacy threats to the users, as IM servers are involved in all user communications. Some IM services deploy end-to-end encryption to alleviate this, as presented below.

Security Features: IM services use standard authentication mechanisms like authorization keys and public key certificates to *authenticate* IM servers and peers [11], [12]. Also, they use standard techniques to ensure the *integrity* of messages. All major IM services encrypt user communications to protect *confidentiality* [34]. Some IM providers additionally deploy *end-to-end encryption* on user communications. This prevents IM operators from seeing the content of communications; however, they can still see communication metadata, e.g., who is talking to whom and when. WhatsApp, Skype, Line, as well as Telegram and Facebook Messenger offer end-to-end encryption, while WeChat, Snapchat, and the BlackBerry Messenger do not.

Also, several major IM providers, including WhatsApp, Viber, Signal, and Facebook Messenger, provide *perfect forward secrecy* by using short-term session keys for user communications [44]. Please refer to Johansen et al. [44] for further discussion of other IM security features.

B. Prior Security Studies of IM Services

Metadata leakage: Coull and Dyer [23] are the first to apply traffic analysis on messaging applications. They demonstrate traffic analysis attacks that can infer various meta-data of a target Apple iMessage user, specifically, the operating system version, type of the IM action, and, to some degree, the language of conversations. More recently, Park and Kim [67] perform traffic analysis on the Korean KakaoTalk IM service, to identify users' online activities using basic classification algorithms. Our work is differ from these works in that the design of our detectors rely on theoretical foundations and meticulous modeling of IM communications. Also, we believe that our attacks are able to reveal IM meta-data that is more sensitive than what was identified by prior works. We demonstrate the applicability of our attacks on several IM services, and design and evaluate tailored countermeasures.

Security vulnerabilities: Johansen et al. [44] surveyed different implementations of SIM protocols such as Signal, WhatsApp, and Threema, and evaluated their security and usability; they conclude that none of the studied applications are infallible. Unger et al. [87] performed a comprehensive study of instant messaging protocols focused on their security properties around trust establishment, conversation security, and transport privacy. Also, Aggarwal et al. [3] study the implementation of encryption in widely-used messaging applications.

Furthermore, there have been various identity enumeration attacks on messaging applications. In particular, as some IM services use SMS text message to activate new devices, an adversarial phone company can initiate and intercept such authorization codes to either identify users or access their accounts. Alternatively, unconfirmed reports [22] suggest that mainland Chinese and Hong Kong authorities may have attempted to discover Hong Kong protesters by misusing a Telegram feature that allowed one to discover the Telegram IDs of phone contacts (therefore, mapping phone numbers to their Telegram IDs); Telegram has promised to fix this issue through an update that will allow users to cloak their phone numbers [75].

Alternatively, Schliep et al. [76] evaluate the security of the Signal protocol against Signal servers. They identify vulnerabilities that allow the Signal server to learn the contents of attachments, re-order and drop messages, and add/drop participants from group conversations. Note that their study targets an entirely different adversary than ours, i.e., their adversary is a compromised/malicious Signal server, whereas in our case the adversary is any third-party who is able to wiretap encrypted IM traffic. Also, their attacks only work against Signal, whereas our attacks apply to all major IM services as they rely on fundamental communication behavior of IM services.

Communication privacy: The centralized nature of popular SIM services makes them susceptible to various privacy issues.

First, all user communications, including group communications and one-on-one communications, are established with the help of the servers run by the SIM providers; therefore, SIM providers have access to the metadata of all communications, i.e., who is talking to whom, and channel ownership and membership relationships. Recent works suggest using various cryptographic techniques, such as private set intersection, to protect privacy against the central operators, e.g., for contact discovery [47]. Second, even if an IM service provider is not malicious, its servers may be compromised by malicious adversaries [30] or subpoenaed by governments, therefore putting client communication metadata at risk.

In traditional SIM services, user communications are encrypted end-to-middle, i.e., between clients and SIM servers. In such services, the SIM providers can see not only the users' communication metadata but also their communication contents. Recently, major SIM providers such as Signal and WhatsApp have started to support end-to-end encryption, therefore protecting communication content from SIM providers. Poor/buggy implementations of some SIM services have resulted in various security flaws and meta-data leakage threats despite their use of end-to-end encryption [32], [49], [60], [74], [94], e.g., through on/off notifications in Telegram [32] and the recent WhatsApp vulnerability giving remote access to the hackers [94].

Censorship: The centralized architecture of popular SIM services makes their censorship trivial: censors can easily blacklist a handful of IP addresses or DNS records to block all communications to a target SIM service. A straightforward countermeasure to unblock censored SIM services is to use standard circumvention systems like VPNs [88] and Tor [27]. Alternatively, major SIM services allow the use of circumvention proxies to evade blocking, e.g., as built into the recent versions of the Telegram software after censorship attempts by Iranian and Russian authorities.

III. Attack and Threat Model

In this work, we demonstrate **a fundamental attack** on IM services: *our attacks are applicable to all major IM services, and are not due to buggy software implementations that can be fixed through software updates*, as overviewed in Section II-B. Our attacks are performed by an adversary who merely performs *traffic analysis*. In this setting, the attacker does not need to compromise or coerce the SIM provider, nor does she need to block the target IM service entirely. Instead, the adversary performs traffic analysis to identify the participants of target IM communications in order to either punish the identified IM participants or *selectively block* the target communications. In particular, the adversary can use traffic analysis to identify the administrators of controversial political or social IM channels and force them to shut down their channels (as seen in recent incidents [1], [2]). Alternatively, the adversary can use our traffic analysis attacks to identify the members of controversial IM channels, and thereby selectively disrupt the access to the target channels.

A. Introducing the Players

The **adversary** is a surveillance organization, e.g., an intelligence agency run by a government. The **goal of the**

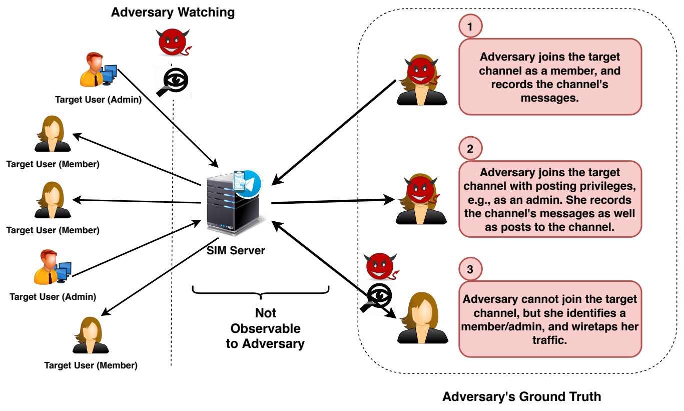

Adversary Watching
Target User (Admin)
Target User (Member)
Target User (Member)
Target User (Admin)
Target User (Member)
SIM Server
Not Observable to Adversary
1
Adversary joins the target channel as a member, and records the channel's messages.
2
Adversary joins the target channel with posting privileges, e.g., as an admin. She records the channel's messages as well as posts to the channel.
3
Adversary cannot join the target channel, but she identifies a member/admin, and wiretaps her traffic.
Adversary's Ground Truth

Fig. 1: Alternative attack scenarios

adversary is to *identify (the IP addresses of) the members or administrators (owners) of target IM conversations*.

A *target IM conversation* can be a public IM channel (e.g., a chat room) on politically or socially sensitive topics, or a private IM conversation between target users, e.g., dissidents and journalists.

For the adversary to be able to conduct the attack, she needs to be *intercepting* the (encrypted) network traffic of the **monitored IM users**, e.g., by wiretapping the ISPs of the monitored users. Therefore, considering the Great Firewall of China as the adversary, she can only perform the attack on the IM users residing inside China.

B. Threat Model

We assume that the hosting IM service is a secure IM (SIM) service, as defined in Section II. Therefore, the adversary does not leverage any security vulnerabilities of the target SIM service in performing the attack. For instance, the SIM system does not leak the IP addresses (or other sensitive meta-data) of its clients to the adversary. Also, we assume all traffic between IM clients and the IM servers to be *encrypted* with strong encryption. Finally, the operators of the SIM service *do not cooperate with the adversary* in identifying target members.

C. How the Attack Is Performed

Figure 1 illustrates the setup of the attack. Suppose that the adversary aims at identifying the members/admins of a specific IM channel, C.

Adversary's ground truth: For any target channel C, the attacker needs to obtain some *ground truth* about the traffic of the channel. This can be done in three ways:

(1) If C is an open (public) channel, the adversary joins the channel (as a member) and records the messages sent on C along with their metadata (e.g., time and size of the messages).
(2) The adversary has joined C and is capable of posting messages to C. This can happen if C a closed group that gives every member the capability to post messages, or this could be because the adversary has gained an admin role for C (e.g., the surveillance adversary has created a channel on a politically sensitive topic to identify target

journalists, or the adversary has arrested the owner of the sensitive channel and is misusing her account). In this setting, not only the adversary can record the messages posted to C, but also he can post his own messages to C with his desired (distinct) traffic patterns.

(3) The adversary is not able to join C as a member/admin, but he has identified (the IP address of) one of the members/admins of C. The adversary wiretaps the (encrypted) network traffic of the identified member and records the traffic patterns of the identified member.

Adversary's wiretap: The adversary monitors the (encrypted) network traffic of IM users to identify (the IP addresses of) the members/admins of the target IM channel C. This can be performed by the adversary wiretapping the network traffic of the ISPs or IXPs he is controlling, e.g., by the Great Firewall of China. Alternatively, the adversary can wiretap the network traffic of specific individuals (e.g., suspected activists), perhaps after obtaining a wiretapping warrant.

Adversary makes decisions: The adversary uses a detection algorithm (as introduced in Section V) to match the traffic patterns of the wiretapped users to the ground truth traffic patterns of the target channel C.

D. Related Traffic Analysis Attacks

Prior work has studied various kinds of traffic analysis attacks in different contexts.

Flow correlation In this setting, the adversary tries to link obfuscated network flows by correlating their traffic characteristics, i.e., packet timings and sizes [25], [28], [37], [51], [59], [78], [93], [98]. Flow correlation has particularly been studied as an attack on anonymity systems like Tor [7], [57], [70], [91], [99]: the adversary can link the ingress and egress segments of a Tor connection (say, observed by malicious Tor guard and exit relays) by correlating the traffic characteristics of the ingress and egress segments. Recently, Nasr et al. [58] introduce a deep learning-based technique called DeepCorr which learns a correlation function to match Tor flows, and outperforms the previous statistical techniques in flow correlation. Alternatively, flow correlation has been studied as a defensive mechanism to detect stepping stone attackers [28], [37].

Flow watermarking This is the active version of flow correlation attacks described above. In flow watermarking, the adversary encodes an imperceptible signal into traffic patterns by applying slight perturbations to traffic features, e.g., by delaying packets [39], [40], [69], [92], [97]. Compared to regular (passive) flow correlation techniques, flow watermarks offer higher resistance to noise, but require real-time modification of network traffic, and are subject to detection attacks.

Website fingerprinting In Website Fingerprinting (WF), the adversary intercepts network connections of some monitored users and tries to match the patterns of the intercepted connections to a set of target webpages. This differs with flow correlation in that flow correlation intercepts the two ends of target connections. WF has particularly been studied as an attack on Tor. Existing WF techniques leverage various machine learning algorithms, such as k-NN, SVM, and deep neural networks to design classifiers that match monitored connections to target web pages [18], [35], [36], [38], [45], [52], [64], [65], [71], [89].

Intersection Attacks Intersection attacks [4], [26], [28], [48] try to compromise anonymous communications by matching users' activity/inactivity time periods. For instance, Kesdogan et al. [48] model an anonymity system as an abstract threshold mix and propose the disclosure attack whose goal is to learn the potential recipients for any target sender. Danezis et al. [24] make the attack computationally more practical by proposing a statistical version of the attack.

Side channel attacks Another class of traffic analysis attacks aims at leaking sensitive information from encrypted network traffic of Internet services [6], [10], [13], [19], [33], [77], [82], [95]. For instance, Chang et al. [19] infer speech activity from encrypted Skype traffic, Chen et al. [20] demonstrate how online services leak sensitive client activities, and Schuster et al. [77] identify encrypted video streams.

Our Traffic Analysis Direction: Our attacks presented in this paper are *closest in nature to the scenario of flow correlation techniques*. Similar to the flow correlation setting, in our scenario the adversary intercepts a live target flow (e.g., by joining a controversial IM channel), and tries to match it to the traffic patterns of flows monitored in other parts of the network (to be able to identify the IP addresses of the members or admins of the target channel). However, we can not trivially apply existing flow correlation techniques to the IM scenario, since the traffic models and communication noise are entirely different in the IM scenario. We, therefore, design flow correlation algorithms tailored to the specific scenario of IM applications. To do so, we first model traffic and noise behavior in IM services, based on which we design tailored flow correlation algorithms for our specific scenario.

Note that one could alternatively use techniques from the intersection attacks literature to design traffic analysis attacks for IM services. However, flow correlation is significantly more powerful than intersection attacks, as flow correlation leverages not just the online/offline behavior of the users, but also the patterns of their communications when they are online. Also, typical IM clients tend to remain online for very long time intervals. Therefore, we expect attacks based on intersection to be significantly less reliable (or require very long observations to achieve comparable reliability) when compared to our flow correlation-based attacks.

IV. Characterizing IM Communications

We start by characterizing IM traffic and deriving a statistical model for it. We will use our model to find analytical bounds for the attack algorithms we design, as well as to generate synthetic IM traffic to be used in some of our experiments.

A. Main IM Messages

IM services allow their users to send different types of messages, most commonly text, image, video, file, and audio messages. IM messages are communicated between users through one of the following major communication forms:

- **Direct messages** are the one-on-one communications between IM users. As mentioned earlier, popular IM services are centralized, therefore all direct messages are relayed through the servers of the IM providers, and unless end-to-end encryption is deployed, the servers can see communication contents.
- **Private (Closed) Group Communications** are communications that happen between multiple users. In groups, every member can post messages and read the messages posted by others. Each group has an administrator member who created the group and has the ability to manage the users and messages. An invitation is needed for a user to join a closed group.
- **Public (Open) Group Communications** which are also called *channels*, are a broadcast form of communication in which one or multiple administrators can post messages, and the members can only read these posts. Users can join channels with no need for an invitation.

Note that some IM services offer other forms of communications, like status messages, that are not relevant to the attacks discussed in our work.

B. Data Collection

In this paper, we collect the bulk of our IM data on the Telegram messaging application. We choose Telegram for two reasons; first, Telegram hosts a very large number of *public* channels that we can join to collect actual IM traffic. This is unlike other popular IM services where most group communications are closed/private. The second reason for choosing Telegram for data collection is that Telegram has been at the center of recent censorship and governmental surveillance attempts [1], [2], [84], [85], as it is home to a multitude of politically and socially sensitive channels. Note that our analysis and attacks are by no means specific to Telegram, as we demonstrate for other messaging services.

We use Telegram's API to collect the communications of 1,000 random channels with different message rates, each for a 24-hours span. For every collected Telegram message, we extract the channel ID it was sent over, its timestamp, the type of message (text, photo, video, audio or file), and the message size. Telegram has a limit of 50 on the number of new channels a user can join every day. Therefore, we use multiple Telegram accounts over several days to perform our data collection (also note that each Telegram account needs to be tied to an actual mobile phone number, limiting the number of accounts one can create).

Although we choose Telegram for collecting SIM traffic, we note that our attack algorithms perform similarly on other SIMs like WhatsApp and Signal. This is because none of these services implement traffic obfuscation, and therefore the shape of their traffic is similar across different IMs. We have illustrated this in Figure 2, where the same stream of messages are sent over four different SIM services. As can be seen, the same messages result in similar traffic patterns across different IM services.

C. Modeling IM Communications

We derive statistical models for IM communications based on our collected IM traffic. We model two key features of IM traffic: inter-message delays (IMDs) and message sizes. We also model the communication latency of IM traffic. We use *Maximum Likelihood Estimation (MLE)* [63] to fit the best probability distribution for each of these features.

TABLE II: Distribution of various message types

Type	Count	Volume (MB)	Size range	Avg. size
Text	12539 (29.4%)	3.85 (0.016%)	1B-4095B	306.61B
Photo	20471 (48%)	1869.57 (0.765%)	2.40Kb-378.68Kb	91.33KB
Video	6564 (15.4%)	232955.19 (95.3%)	10.16Kb-1.56Gb	35.49MB
File	903 (2.1%)	47.46 (0.019%)	2.54Kb-1.88Mg	52.56KB
Audio	2161 (5.1%)	9587.36 (3.92%)	2.83Kb-98.07Mg	4.44MB

Inter-Message Delays (IMDs): The IMD feature is the time delay between consecutive IM messages in an IM communication. In our model, we merge messages sent with very small IMDs (specifically, messages separated by less than a threshold, t_e seconds). We do this because such extremely close messages create a combined traffic burst in the encrypted IM traffic that cannot be separated by the traffic analysis adversary. Such close messages appear (infrequently) when an administrator forwards a batch of IM messages from another group. We also filter out the very long IMDs that correspond to long late-night inactivity periods.

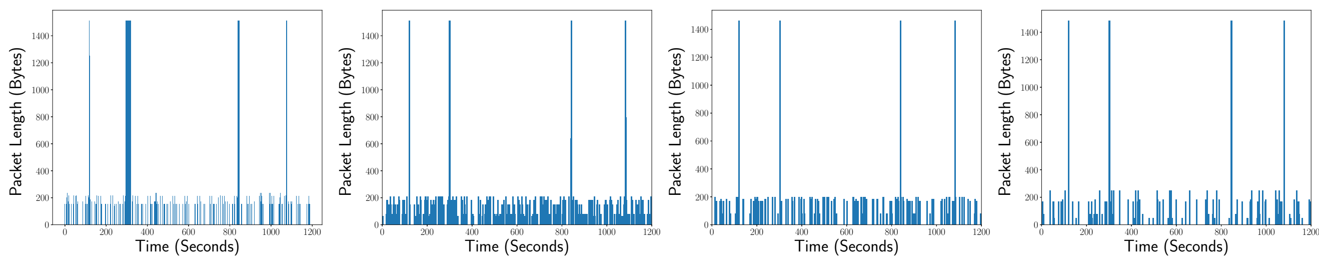
We show that the probability density function of IMDs can be closely fitted to an exponential distribution using our MLE algorithm; Figure 3 shows the probability density function of IMDs for 200 IM channels with a message rate of 130 messages per day. We interpret the exponential behavior of the IMDs to be due to the fact that messages (or message batches) are sent independently in the channels (note that this will be different for interactive one-on-one chats).

Also, we consider IMDs to be independent of the type and size of messages, since in practice there is no correlation between the time a message is sent and its type or size.

Messages Sizes: Table II shows the size statistics and frequencies of the five main message types in our collected IM messages. We use these empirical statistics to create a five-state Markov chain, shown in Figure 6, to model the sizes of the messages sent in an IM communication stream. We obtain the empirical transition probability matrix of this Markov model for the aggregation of all channels, as well as for groups of channels with similar rates; the matrices are presented in Appendix D. We see that the transition matrices change slightly for IM channels with different daily message rates.

Finally, Figure 4 shows the Complementary Cumulative Density Function (CCDF) of the normalized message sizes for different message types (the sizes are normalized by the maximum message size of each category). We observe that different message types are characterized by different message size distributions.

Communication Latency: IM messages are delayed in transit due to two reasons: network latency and the IM servers' processing latency. To measure such latencies, we collect IM traffic from 500 channels, each for one hour (therefore, 500 hours worth of IM traffic) using Telegram's API. We then set up two IM clients, and send the collected IM traffic between the two clients to measure the incurred communication latencies. Using MLE, we find that transition latencies fit best to



(a) Telegram (b) Signal (c) WhatsApp (d) Viber

Fig. 2: Comparing the shape of traffic on four major SIM services; by sending the same sequence of IM messages, we observe similar traffic bursts regardless of the service provider.

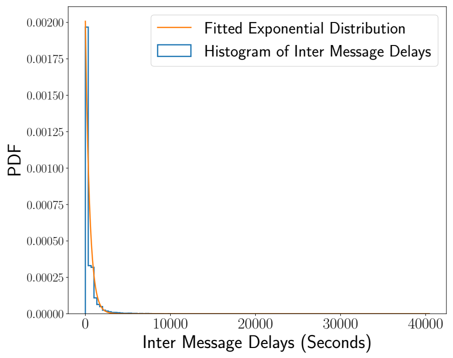


Fig. 3: The PDF of inter-message delays and its fitted exponential distribution

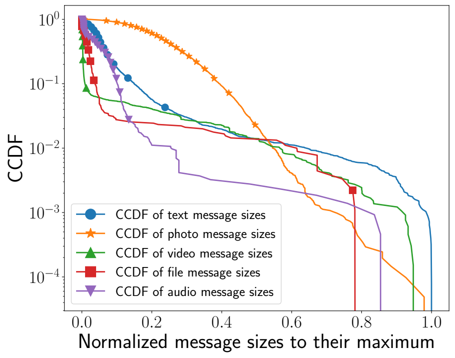


Fig. 4: Complementary CDF (CCDF) of IM Size distributions for different types of messages

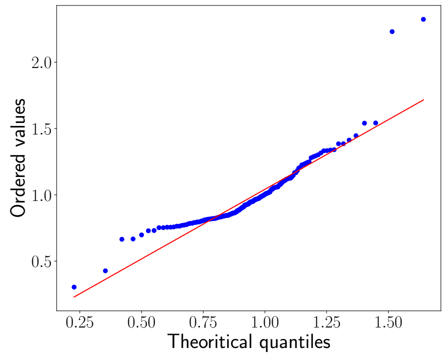


Fig. 5: The Quantile-Quantile plot of transition delay and its fitted Laplacian distribution

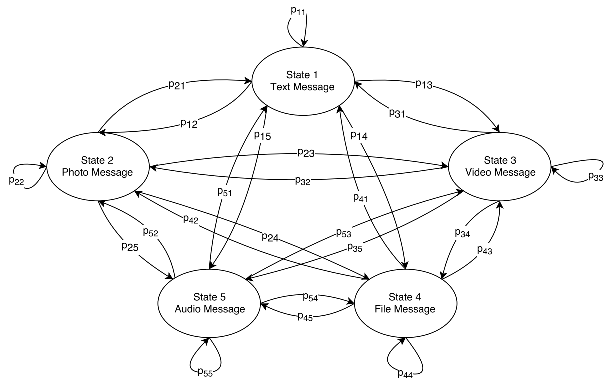


Fig. 6: Markov chain of IM message sizes

a Laplacian distribution $f_{\mu,b}(x)$, where μ is the average and $2b^2$ is the variance of the delay. Since network delay cannot be negative, we consider only the positive parts of the Laplace distribution. Figure 5 shows a Quantile-Quantile (Q-Q) plot of the packet latencies against the best Laplace distribution.

D. Synthesizing SIM Traffic

We can use our empirical model, described above, to create synthetic IM communications. A synthetic IM communication trace consists of IM messages with specific sizes, timings, and message types. Such synthetic traffic allows us to validate our experimental results using much more traffic samples than what can be collected online.

The sketch of our algorithm in presented in Algorithm 1. The inputs of the algorithm are λ, the rate of messages (per day) for the channel to be synthesized, and T, the length of the synthesized channel. First, the algorithm uses the empirical distribution of IMDs (shown in Figure 3) to create a sequence of IM message timings. Then, our algorithm uses our Markov model for message types to assign a type to each of the messages in the sequence (Section IV-C). Finally, for each message, our algorithm finds its size using the empirical distribution of sizes for the corresponding type of the message (i.e., from Figure 4). The output of the algorithm is a sequence of IM messages, where each message has a timestamp, a size, and a message type.

Later in Section VI-E we show that our detectors provide comparable performances on synthetic and real-world IM traces (for similar settings), demonstrating the realisticity of our synthetic traces. Since our traffic synthesizing algorithm uses sample IM traces to generate synthetic IM traffic patterns, the quality of its synthesized traffic improves by increasing the size of its training dataset. Alternatively, one could train a generative adversarial network to produce synthetic IM traces; we leave this to future work.

V. Details of Attack Algorithms

We design two algorithms for performing our attack (i.e., to map monitored IM users to their channels). As discussed in Section III-D, our attack scenario is closest in nature

Algorithm 1 Algorithm for Generating Synthetic IM Traffic

```
1:  procedure GenerateSyntheticTraffic
2:      $P \leftarrow$ transition matrix based on $\lambda$
3:      $current\ length \leftarrow 0$
4:      $message\ sequence \leftarrow \varnothing$
5:      while $current\ length < T$ do
6:          $event \leftarrow \varnothing$
7:          $t \leftarrow$ A random time from IMD
8:              empirical distribution
9:          $current\ length \leftarrow current\ length + t$
10:         $event \leftarrow event + \{t\}$
11:         $message\ sequence \leftarrow message\ sequence + \{event\}$
12:     for each $event$ in $message\ sequence$ do
13:         $event \leftarrow event + \{message\ type\}$ based on $P$
14:         $size \leftarrow$ A random size from the corresponding
15:             message type empirical distribution
16:         $event \leftarrow event + \{size\}$
17:     return $message\ sequence$
```

to the scenario of flow correlation attacks. Therefore, the design of our attacks is inspired by existing work on flow correlation. Prior flow correlation techniques use standard statistical metrics, such as mutual information [21], [100], Pearson correlation [50], [78], Cosine Similarity [41], [59], and the Spearman Correlation [81], to link network flows by correlating their vectors of packet timing and sizes. We use *hypothesis testing* [68],[2] similar to state-of-the-art flow correlation works [40], [41], to design optimal traffic analysis algorithms for the particular setting of IM communications. In contrast to flow correlation studies which use the features of network packets, we use the features (timing and sizes) of *IM messages* for detection.

Note that, the recent work of DeepCorr [58] uses a deep learning classifier to perform flow correlation attacks on Tor. They demonstrate that their deep learning classifier outperforms statistical correlation techniques in linking Tor connections. In Section VI-D, we compare our statistical classifiers with a DeepCorr-based classifier tailored to IM traffic. As we will show, our statistical classifiers *outperform* such deep learning based classifiers, especially for shorter flow observations. Intuitively, this is due to the sparsity of events in typical IM communications, as well as the stationary nature of noise in IM communications in contrast to the scenario of Tor. Note that this fully complies with Nasr et al. [58]'s observation that DeepCorr only outperforms statistical classifiers in non-stationary noisy conditions, where statistical traffic models become inaccurate.

Our hypothesis testing: Consider C to be a target SIM channel (e.g., a public group on a politically sensitive topic). For each IM user, U, the attacker aims at deciding which of the following hypotheses is true:

- H_0: User U is *not* associated with the target channel C, i.e., she is neither a member nor an administrator of channel C.

[2]Our approach is “threshold testing” by some of the more strict definitions, however, we will use the term “hypothesis testing” in this paper as threshold testing falls into the general class of statistical hypothesis tests [68].

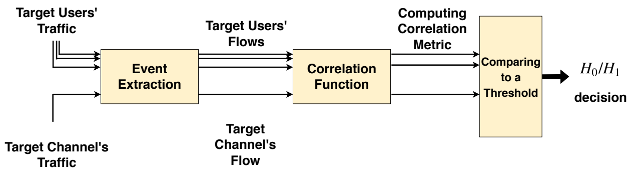


Fig. 7: Event-based detector

- H_1: User U *is* associated with the target channel C, i.e., she is posting messages to that channel as an admin, or is a member of that channel and therefore receives the channel's messages.

As described in our threat model (Section III), the adversary can only observe encrypted SIM communications between users and SIM servers. Therefore, we design detectors that use traffic features, i.e., IMDs and message sizes. In the following, we describe two detector algorithms.

A. Event-Based Detector

Our first detector, the *Event-Based Detector*, aims at matching SIM events in a target user's traffic to those of the target channel C. An *event* $e = (t, s)$ is a single SIM message or a batch of SIM messages sent with IMDs less than a threshold t_e (as introduced earlier). Each single SIM message can be one of the five types of image, video, file, text, or audio. t is the time that e appeared on the SIM communication (e.g., sent to the public channel), and s is the size of e. Note that an SIM communication can include SIM protocol messages as well (handshakes, notifications, updates, etc.); however, such messages are comparatively very small as shown in Figure 8, and thus the detector ignores them in the correlation process. Recall that the adversary is not able to see plaintext events in the user's traffic due to encryption. Therefore, the first stage of our event-based detector is to *extract* events based on the user's encrypted SIM traffic shape. Figure 7 depicts the components of our event-based detector.

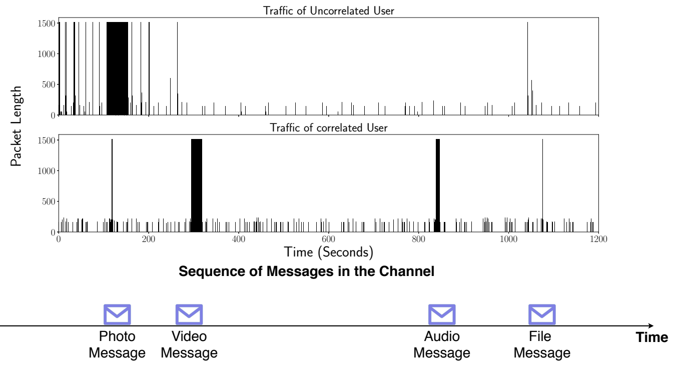


Fig. 8: Event extraction: IM Messages sent/received by a target user create bursts of (encrypted) packets; the adversary can extract events from packet bursts.

Event Extraction: Each SIM event, e.g., a sent image, produces a burst of MTU-sized packets in the encrypted traffic, i.e., packets with very small inter-packet delays. This is illustrated in Figure 8: SIM events such as images appear as traffic bursts, and scattered packets of small size are SIM protocol messages like notifications, handshakes, updates, etc. Therefore, the adversary can extract SIM events by looking for bursts of MTU-sized packets, even though she cannot see packet contents due to encryption. We use the IPD threshold t_e to identify bursts. Any two packets with distance less than t_e are considered to be part of the same burst. Note that t_e is a hyper-parameter of our model and we discuss its choice later in the paper. For each burst, the adversary extracts a SIM event, where the arrival time of the last packet in the burst gives the arrival time of the event, and the sum of all packet sizes in the burst gives the size of the event. Two SIM messages sent with an IMD less than t_e are extracted as one event. Similarly, the adversary combines events closer than t_e when capturing them from the target channel.

Forming Hypotheses: We call a one-sided SIM communication a SIM *flow*. Therefore, a flow either consists of the packets *sent* by a user to an SIM server, or the packets *received* by the user from the SIM server. We represent a flow with n events as $f = \{e_1, e_2, \ldots, e_n\}$, where $e_i = (t_i, s_i)$ is the ith event.

Consider a user U and a target channel C. Suppose that the adversary has extracted flow $f^{(U)} = \{e_1^{(U)}, e_2^{(U)}, \ldots, e_n^{(U)}\}$ for user U (through wiretapping), and flow $f^{(C)} = \{e_1^{(C)}, e_2^{(C)}, \ldots, e_n^{(C)}\}$ for the target channel C (using her ground truth). The detector aims at deciding whether user U is an administrator or member of the channel. We can re-state the adversary's hypotheses presented earlier in this section as follows:

- H_0: User U is not an administrator or member of the target channel; hence, $f^{(C)}$ and $f^{(U)}$ are independent.
- H_1: User U is an administrator or member of the target channel C; therefore, the user flow $f^{(U)}$ is a noisy version of the channel flow $f^{(C)}$.

Therefore, we have[3]

$$\begin{cases} H_0 : t_i^{(C)} = t_i^{(*)} + d_i^{(*)}, s_i^{(C)} = s_i^{(*)}, 1 \le i \le n \\ H_1 : t_i^{(C)} = t_i^{(U)} + d_i^{(U)}, s_i^{(C)} = s_i^{(U)}, 1 \le i \le n \end{cases}$$

where $f^{(*)} = \{e_1^{(*)}, e_2^{(*)}, \ldots, e_n^{(*)}\}$ is the flow of a user $U' \neq U$ who is *not* an administrator/member of channel C. Also, $d_i^{(\cdot)}$ is the latency applied to the timing of the ith event. Note that IM message sizes do not change drastically in transit, and the order of messages remains the same after transmission.

Detection Algorithm: The adversary counts the number of event matches between the user flow $f^{(U)}$ and the channel flow $f^{(C)}$. We say that the ith channel event $e_i^{(C)}$ matches *some* event $e_j^{(U)}$ in $f^{(U)}$ if:

- $e_i^{(C)}$ and $e_j^{(U)}$ have close timing: $|t_i^{(C)} - t_j^{(U)}| < \Delta$; and
- $e_i^{(C)}$ and $e_j^{(U)}$ have close sizes: $|s_i^{(C)} - s_j^{(U)}| < \Gamma$.

where Δ and Γ are thresholds discussed in Section VI-A. Note that even though the sizes of SIM messages do not change in transmission, the event extraction algorithm introduced earlier may impose size modifications, as network jitter is able to divide/merge event bursts (i.e., a burst can be divided into two bursts due to network jitter or two bursts can be combined due to the small bandwidth of the user).

[3]Note that the above hypothesis is for the case that the adversary is looking for the "administrator" of channel C. For the case that the adversary is looking for the "members" of the target channel, the hypothesis changes slightly by replacing $t_i^{(C)} = t_i^{(U)} + d_i^{(U)}$ with $t_i^{(U)} = t_i^{(C)} + d_i^{(C)}$. As the derivations will be exactly the same, we exclude it without loss of generality.

Finally, the adversary calculates the ratio of the matched events as $r = k/n$, where k is number of matched events and n is the total number of events in the target channel. The detector decides the hypothesis by comparing to a threshold: $r = \frac{k}{n} \underset{H_0}{\overset{H_1}{\gtrless}} \eta$. The detection threshold η is discussed in Section VI-B.

Analytical Bounds: We first derive an upper-bound on the probability of false positive ($\mathbb{P}_{\text{FP}}$), i.e., the probability that H_1 is detected when H_0 is true (Type I error). Let p_0 be the probability that a message with size $s_i^{(C)}$ and time $t_i^{(C)}$ matches an event in $f^{(U)}$ when H_0 is true, i.e., there exists only one message whose time $t_j^{(*)}$ satisfies $t_i^{(C)} \le t_j^{(*)} \le t_i^{(C)} + \Delta$ and has the same size label as $s_i^{(C)}$. From our observations, $p_0 = 0.002$. This Type I error occurs if more than $\eta \cdot n$ events in $f^{(C)}$ match $f^{(U)}$, when H_0 is true. This is equivalent to the case that less than $n - \eta \cdot n$ events in $f^{(C)}$ do not match $f^{(U)}$ when H_0 is true. Consequently,

$$\begin{aligned} \mathbb{P}_{\text{FP}} = \mathbb{P}(k \ge \eta n \mid H_0) &= \mathbb{P}(n - k \le n - \eta n \mid H_0), \\ &= F(n - \eta n; n, 1 - p_0), \\ &\le \left(\frac{1-\eta}{p_0}\right)^{-n+n\eta\eta} \left(\frac{\eta}{1-p_0}\right)^{-n\eta}, \end{aligned} \tag{1}$$

where $F(r; m, p) = \mathbb{P}(X \le r)$ is the cumulative density function of a Binomial distribution with parameters m, p, and the last step follows from the following inequality which is tight when p is close to zero [5]:

$$F(r; m, p) \le \left(\frac{r/p}{p}\right)^{-k} \left(\frac{1 - r/m}{1-p}\right)^{k-m} \tag{2}$$

Next, we upper-bound the probability of false negatives ($\mathbb{P}_{FN}$), i.e., the probability that H_0 is detected when H_1 is true, which occurs when less than k messages of $f^{(C)}$ match $f^{(U)}$. Let p_1 be the probability of the case that an event of $f^{(C)}$ matches $f^{(U)}$ when H_1 is true (Type II error).

Even though we mentioned earlier in this section that when H_1 is true, a delayed version of each event of $f^{(U)}$ appears in $f^{(C)}$, the bandwidth of the target user can affect the burst extraction process. As explained earlier in this section, we merge bursts of packets for messages whose IMD is less than t_e. Hence, suppose that the time it takes for the user to send a message is large enough to make the IMD between the current message and the next one less than t_e. Therefore, these two consecutive messages are combined in one burst. Table III shows the value of p_1 observed from our data for different bandwidths. Since the bandwidth of our experiments is 1Mbps, $p_1 = 0.921$.

Note that Type II error occurs when less than $\eta \cdot n$ messages of $f^{(C)}$ match $f^{(U)}$ when H_1 is true. Therefore,

$$\begin{aligned} \mathbb{P}_{\text{FN}} = \mathbb{P}(k \le \eta n | H_1) &= F(\eta n; n, p_1) \\ &\le \left(\frac{\eta}{p_1}\right)^{-n\eta} \left(\frac{1-\eta}{1-p_1}\right)^{\eta n - n}, \end{aligned} \tag{3}$$

where the last step follows from (2).

TABLE III: The empirical value of p_1 measured for different client bandwidths

Client Bandwidth (Mbps)	p_1
0.1	0.824
0.5	0.902
1	0.921
10	0.974
100	0.983

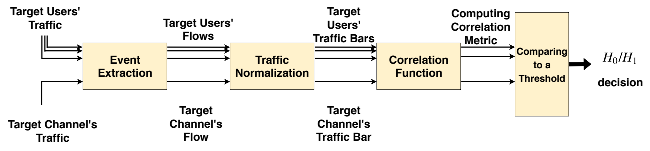


Fig. 9: Shape-based detector

We will validate the upper-bounds of $\mathbb{P}_{\text{FP}}$ and $\mathbb{P}_{\text{FN}}$ with our live experiments (Section VI-B) and simulation results (Section VI-E).

B. Shape-Based Detector

We design a second detector called the *shape-based* detector. This detector links users to SIM communications by correlating the shape of their network traffic, where traffic shape refers to the vector of packet lengths over time. Figure 9 illustrates the four stages of the shape-based detector.

Event Extraction: The first stage of the shape-based detector is to extract SIM events from network traffic, which is performed similar to what was described earlier for the event-based detector. As described in the following, we do this in a way that accounts for the different bandwidths of the users being correlated.

Normalizing Traffic Shapes: The shape-based detector converts the extracted events into normalized traffic shapes by replacing each event with a traffic bar. The reason for doing so is that the shape of an IM event (e.g., the corresponding packet burst) is a function of user network bandwidths; our traffic normalization removes the impact of user bandwidth, and therefore the adversary can correlate traffic shapes with no knowledge of the underlying users' bandwidths.

To perform this normalization, we replace each event (i.e., each burst) with a traffic bar whose width is $2 \times t_e$, where t_e is the threshold used during event extraction as discussed in section V-A. We choose this value to reduce the chances of overlaps between consecutive bars. To capture the sizes of events in traffic normalization, the height of each bar is chosen such that the area under the bar equals the size of the event. Our shape normalization also reduces correlation noise by removing small traffic packets that are not part of any SIM events.

To form the new normalized shape of traffic, we divide each bar into smaller bins of width t_s, the value of which is discussed in Section VI-A, and with a height equal to the height of the corresponding bar. Therefore each bar consists of a number of bins of equal width and height. Furthermore, we put bins with the same width t_s and height 0 between these bars. By doing so, after the traffic normalization, the new shape of traffic will be the vector of heights of bins over time.

Correlating Normalized Traffic Shapes: Our shape-based detector correlates the normalized shapes of two traffic streams of target channel C and user U to decide if they are associated. Suppose that $b^{(C)} = \{b_1^{(C)}, b_2^{(C)}, \ldots, b_{n_C}^{(C)}\}$ and $b^{(U)} = \{b_1^{(U)}, b_2^{(U)}, \ldots, b_{n_U}^{(U)}\}$ are the respective vectors of heights of bins associated with the target channel and user being tested, where n_C and n_U are the number of events in target channel and user flows, respectively. We use the following normalized correlation metric:

$$corr = 2 \times \frac{\sum_{i=1}^{n} b_i^{(C)} b_i^{(U)}}{\sum_{i=1}^{n} (b_i^{(C)})^2 + \sum_{i=1}^{n} (b_i^{(U)})^2} \tag{4}$$

where $n = \min(n_C, n_U)$. Note that $corr$ returns a value between 0 and 1, which shows the similarity of the two traffic shapes (1 shows the highest similarity). Finally, the detector makes its decision by comparing $corr$ to a threshold, $corr \underset{H_0}{\overset{H_1}{\gtrless}} \eta$, where η is the detection threshold.

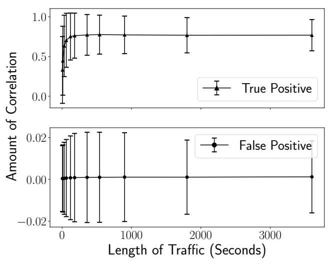
VI. Attack Experiments

A. General Setup

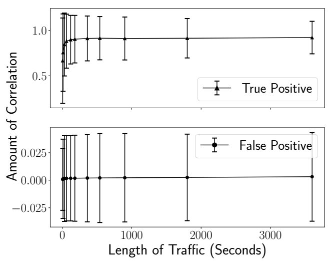
We design our experimental setup to perform our attacks in the setting of Figure 1, and based on the threat model of Section III. We use the first type of ground-truth in Figure 1 (adversary joins the target channel as a reading-only member), which is more challenging for the adversary compared to the second ground-truth, and similar in performance compared to the third ground-truth mechanism. Specifically, we use two SIM clients using different SIM accounts (e.g., Telegram accounts) that are running IM software on two separate machines. One of these IM clients is run by the adversary, and the other one represents the target client. The adversary client joins target channel C, (e.g., a public political Telegram channel) and records the metadata of all the SIM communications of C, i.e., the timing and sizes of all messages sent on that channel. The target client may or may not be a member/admin of the target channel C. The adversary is not able to see the contents of the target client's communications (due to encryption), however she can capture the encrypted traffic of the target client. The adversary then uses the detection algorithms introduced in Section V to decide if the target user is associated with the target channel C. In a real-world setting, the adversary will possibly have multiple target channels, and will monitor a large number of suspected clients.

For our Telegram and Signal experiments, our adversary-controlled client uses their APIs to record SIM communications of target channels, while for WhatsApp, we manually send messages through its Desktop version (as it does not have an API).

Parameter Selection. We choose burst detection threshold as $t_e = 0.5s$ based on the empirical distribution of network jitter. Also, we set t_s of the shape-based detector to $0.01s$, as it leaves enough separation between two consecutive IM messages. We set Δ of the event-based algorithm to 3 seconds. We also set Γ parameter of the event-based detector to $10Kb$.



(a) Shape-based detector



(b) Event-based detector

Fig. 10: Attack correlation metrics for different traffic lengths. The performance improves for longer observations.

Ethics. We performed our inference attacks only over public IM channels; therefore, we did not capture any private IM communications. Also, we performed our attacks only on our own IM clients, but no real-world IM clients. Therefore, our experiments did not compromise the privacy of any real-world IM members or admins.

B. In-The-Wild Attacks on Telegram

We experiment our attacks on in-the-wild Telegram channels (fully complying with the ethical considerations of Section VI-A).

Experimented Channels As discussed in Section IV-B, SIM services put limits on the number of channels a client can join. For our experiments, we joined 500 channels popular among Iranian users, with different rates of daily messages. In our experiments, each time our own client connects to one of these 500 channels, and the goal of the adversary is to match our client to the channel she has joined.

Synchronization As the adversary's clock may be skewed across her vantage points, our adversary uses a simple sliding window to mitigate this: for the first 10 seconds of traffic, the adversary slides the two flows being compared with 0.5 second steps, and uses the maximum correlation value.

Choosing the Threshold Figure 10 shows the TP and FP rates of our experiments for different detection thresholds η, and for different traffic lengths (traffic length excludes the long inactivity periods across correlated connections). Each point in the graph shows the average of the correlation metric across all experiments, and the bars show the standard deviation. Intuitively, the detector performs well for wider gaps between the TP and FP bars. From the figures, we see the impact of the traffic length on detection performance: *longer observations improve detection performance* of our attacks. Also, η trades off the TP and FP rates. The adversary can detect the right threshold and the right traffic length based on her target TP and FP values.

Comparing our Two Attacks: Figures 12 and 13 show the performance of event-based and shape-based detectors, respectively, for different detection thresholds using an ROC curve (for 4 different observation lengths). We can see that, as expected, *longer traffic observations improve the accuracy of both detectors*. For instance, the shape-based detector offers a TP $=$ .94 and FP $= 10^{-3}$ with 15 min observation, while an hour of observation reduces the FP to close to FP $= 5 \times 10^{-5}$. In practice, *an adversary can deploy the attack with hierarchical observation intervals to optimize accuracy and computation*. For instance, the adversary can monitor a mass of IM users for 15 mins of observation; then the adversary will monitor only the clients detected with 15 mins observations for longer time periods, e.g., an hour, to improve the overall FP performance while keeping computations low.

TABLE IV: Comparing the runtimes of our two attacks.

Method	One correlation time
Shape-based correlation	$167\mu s$
Event-based correlation	$2\mu s$

Furthermore, as can be seen, *the shape-based detector outperforms the event-based detector for smaller values of false positive rates*. For instance, for a target true positive rate of 0.9, the shape-based detector offers a false positive of 5×10^{-4} compared to 8×10^{-4} of the event-based detector (with 15 mins of observation). The performance gap decreases for higher false positive rates. The reason for this performance gap is the impact of event extraction noise on the event-based detector. Such noise has smaller impact on the shape-based detector as it correlates the shape of traffic flows.

On the other hand, *our event-based detector is two orders of magnitude faster than the shape-based detector*. Table IV compares the correlation times of the two detectors (averaged over 100 experiments). The main reason for this difference is that the event-based correlator uses the discrete time-series of event metadata for its correlation, while the shape-based detector uses traffic histograms over time.

Note that for our event-based detector in Figure 12, for short traffic observations (e.g., 15 mins) we cannot observe small FPs in our ROC curve. This is because the event-based correlation uses the number of matched events, which is very coarse-grained due to the limited number of events in short (e.g., 15 minutes) intervals. We use our analytical upper-bounds (derived in (1) and (3)) to estimate the performance trend for smaller false positive values (Figure 22 of Appendix).

C. In-The-Wild Attacks on WhatsApp and Signal

As discussed previously in Section IV-B, we make the bulk of our data collection and experiments on Telegram due to its huge number of public channels (making our experiments ethical and realistic as we do not need to do experiments on private communications). However, as shown in Figure 2, the shape of traffic is similar across different SIM services, and therefore we expect our attack algorithms to perform similarly when applied by an adversary on other SIM applications. We validate this through experiments on Signal and WhatsApp messengers.

Signal and WhatsApp only offer private (closed) channels. For ethical reasons, we make our own (closed) channels on these services to perform our experiments. Specifically, we create a private channel on each of Signal and WhatsApp. We send messages on these channels by mimicking the patterns (i.e., inter-message times and message sizes) of randomly chosen public Telegram channels. Our user and adversary VMs join these channels, and we perform our attacks in the same setting as the Telegram experiments.

Figure 14 shows the performance of event-based and shape-based detectors in both Signal and WhatsApp applications using 15 minutes of traffic, *demonstrating detection performances comparable to those of Telegram*. In particular, similar to Telegram, for smaller false positive rates, the shape-based detector has better performance while the event-based detector achieves more accuracy for larger false positive rates. Our results show that **our attacks generalize to SIMs other than Telegram due to the similarity of their traffic patterns**, which is due to these services not using any obfuscation mechanisms.

D. Comparison with Deep Learning Techniques

As mentioned earlier in Section III-D, the recent work of DeepCorr [58] uses deep learning classifiers to perform flow correlation attacks on Tor. They demonstrate that deep learning classifiers outperform statistical correlation techniques, like the ones we used in our work, in correlating Tor connections. In this section, we compare our IM classifiers with deep learning classifiers. As we show in the following, *our statistical classifiers outperform deep learning-based classifiers*, specially for shorter flow observations. Intuitively, this is due to the sparsity of events in typical IM communications, as well as the stationary nature of noise in IM communications in contrast to the scenario of Tor. Note that this fully complies with Nasr et al. [58]'s observation that DeepCorr only outperforms statistical classifiers in non-stationary noisy conditions, where statistical traffic models become inaccurate.

For fair comparisons, we obtain the original code of DeepCorr [58], and adjust it to the specific setting of IM traffic. Specifically, we divide the timing of each flow to equal periods of length 1 second, and in each period we assign values of $\{0, 1\}$ to that period. We set the value of a period 1 if there is a burst of packets in that period, and 0 if there is no burst of packets. As an example, if we use 15 minutes of traffic flows for correlation, our feature dimension is a 900-length vector with values of $0, 1$. We train our classifier using 500 associated flow pairs and $2,000$ non-associated flow pairs. We test our (DeepCorr-based and statistical) classifiers using a non-overlapping set of 200 associated flow pairs and $4,000$ non-associated.

Figure 11 shows the ROC curves of our event-based detector compared with our deep learning-based detector, using 3 and 15 minutes of traffic. As we can see, *our event-based technique outperforms the deep learning-based classifier for smaller false positive rates*. For instance, for a false positive rate of 10^{-3} and using 15 minutes of traffic, our event-based detector achieves a 93% accuracy compared to 88% of the DeepCorr-based technique. We see that the performance advantage of our event-based detector significantly increases for shorter flow observations, e.g., when 3 minutes of traffic is used for detection, our classifier provides 92% accuracy compared to 45% of the DeepCorr-based classifier (for the a false positive rate of 10^{-3}).

E. Simulations Using Synthetic Traffic

Our in-the-wild experiments, presented above, have been done on a limited number of SIM channels, which is due to the

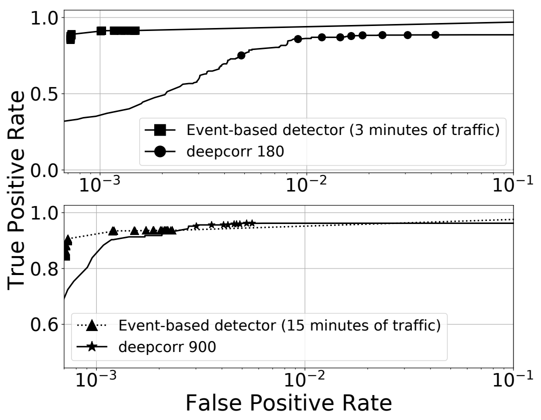


Fig. 11: Comparing our event-based detector with a DeepCorr-based classifier, for 3 and 15 minutes of traffic observation

fact that major SIM services put limits on the number of channels a client can join. To ensure the reliability of our results, we generated a large number of synthetic SIM channels (using the algorithm from Section IV-D), and evaluated our attack performance on them. *Our evaluations using 10,000 synthetic IM channels complies with our results from our in-the-wild experiments*. We have presented our synthetic evaluations in Appendix B.

F. Discussions

1) Impact of Other Channels: In our experiments, a target user is the member/admin of only a single target channel, while in practice a user may be a member/admin of multiple channels. Therefore, a valid question is whether the traffic patterns of other channels may interleave with the patterns of the target channel and reduce the reliability of our detectors. We argue that this will not be an issue as long as the detection is performed during the time interval the user is visiting/posting to the target channel. This is because when a user is visiting a target channel, he will not receive the messages sent to other channels (he will only receive some small-sized notifications). Also, if an admin user simultaneously sends a message to multiple channels, his upstream traffic (to the IM server) will only contain a single message. Therefore, to identify whether a given user is the member/admin of a target channel, the adversary needs to continuously monitor that user until the user visits or posts to the target channel.

2) Impact of Network Conditions: While we have performed our experiments in specific network conditions, we argue that our detectors will perform equivalently in other network conditions as well. This is because in our presented threat model, the adversary has knowledge on the network conditions of each target user (e.g., the adversary can be the ISP of the target user), and therefore she can adjust the detector for various users, as shown in Table III. Also, note that natural variations in a user's network conditions will not impact the detectors as IM traffic patterns are resilient to natural network perturbations (e.g., the distance between IM bursts as shown in Figure 2 are orders of magnitude larger than network jitter).

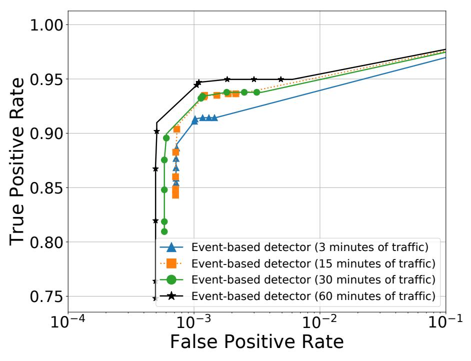


Fig. 12: Performance of the event-based detector over in-the-wild Telegram traffic

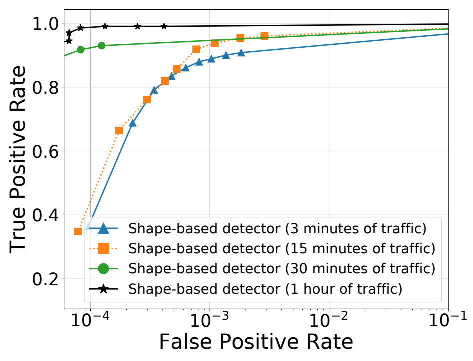


Fig. 13: Performance of the shape-based detector over in-the-wild Telegram traffic

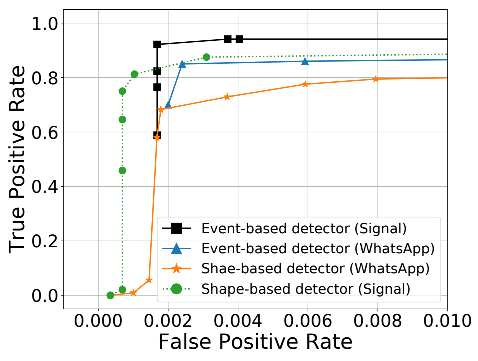


Fig. 14: Performance of event-based and shape-based detector on Signal and WhatsApp

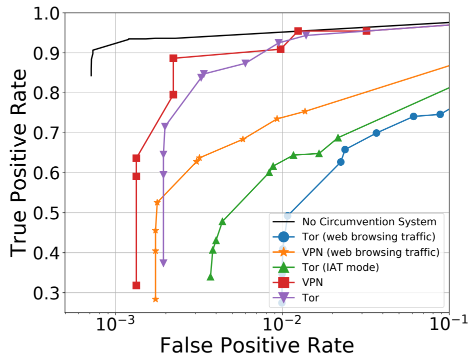


Fig. 15: The impact of various countermeasures on the performance of the event-based detector using different circumvention systems (15 minutes of observed traffic)

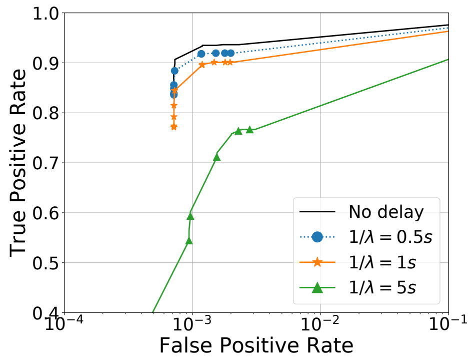


Fig. 16: Randomly delaying events by an SIM server acts as an effective countermeasure to our attacks. $\frac{1}{\lambda}$ is the mean of the added delay (15 minutes of observed traffic)

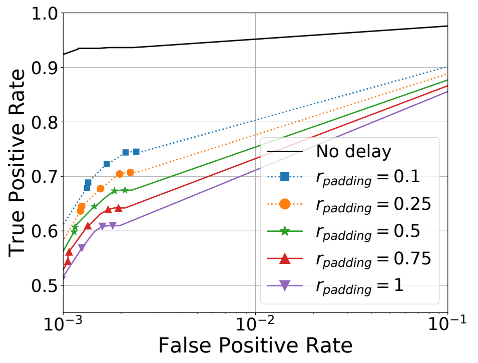


Fig. 17: Padding IM events by the SIM server (or client) can act as an effective countermeasure against our attacks. (15 minutes of observed traffic)

VII. Countermeasures

We deploy and evaluate possible countermeasures against our presented attacks. Intuitively, our attacks work because in-the-wild SIM services do not deploy any mechanisms to obfuscate traffic patterns. Therefore, we investigate various traffic obfuscation mechanisms as countermeasures against our traffic analysis-based attacks.

Note that obfuscation-based countermeasures have been studied against other kinds of traffic analysis attacks overviewed in Section III-D. There are several key ideas used in existing countermeasures: (1) tunneling traffic through an overlay system that perturbs its patterns [54], [62], e.g., Tor, (2) adding background traffic (also called decoy) that is mixed with the target traffic [29], [53], [66], [90], [96], (3) padding traffic events (e.g., packets) [16], [17], [29], [46], [89], and (4) delaying traffic events [16], [17], [29], [89], [90]. In the following, we investigate various countermeasure techniques inspired by these standard approaches.

A. Tunneling Through Circumvention Systems With/Without Background Traffic

As the first countermeasure, we tunnel SIM traffic through standard circumvention systems, in particular VPN and Tor pluggable transports [86]. We use the same experimental setup as before and connect to 300 Telegram channels. For each circumvention system, we perform the experiments with and without any background traffic. In the experiments with background traffic, the VM running the SIM software also makes HTTP connections using Selenium. The background HTTP webpages are picked randomly from the top 50,000 Alexa websites. To amplify the impact of the background traffic, the time between every two consecutive HTTP GETs is taken from the empirical distribution of Telegram IMDs, therefore producing a noise pattern similar to actual SIM channels.

We observe that our event-based attack performs stronger against our countermeasures. Therefore, we only present the countermeasure results against the event-based detector (we show the results for the shape-based attack in Appendix A). Figure 15 shows the ROC curve of the event-based detector using various circumvention systems and in different settings.

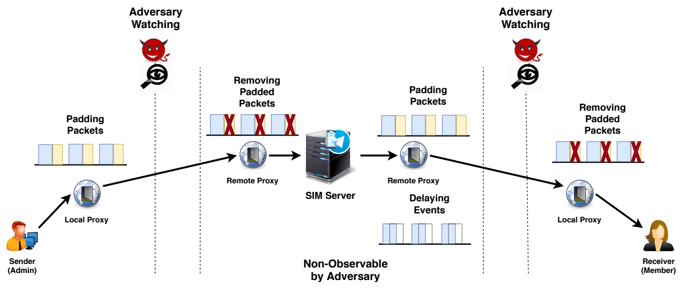


Fig. 18: Design of our IMProxy countermeasure.

Our Tor experiments are done once with regular Tor, and once using the obfs4 [62] transport with the IAT mode of 1, which obfuscates traffic patterns.

We see that *using regular Tor (with no additional obfuscation) as well as using VPN does not significantly counter our attacks*, e.g., we get a TP of 85% and a FP of 5×10^{-3} when tunneling through these services (using 15 mins of traffic). However, adding *background traffic* when tunneled through Tor and VPN reduces the accuracy of the attack, but *we get the best countermeasure performance using Tor's obfs4 obfuscator*.

Note that tunneling through a generic circumvention system like Tor is not the most attractive countermeasure to the users due to the poor connection performance of such systems.

B. IMProxy: An Obfuscation Proxy Designed for IM Services

We design and implement a proxy-based obfuscation system, called IMProxy[4], built specifically for IM communications. IMProxy combines two obfuscation techniques: changing the timing of events (by adding delays), and changing the sizes of events through adding dummy traffic. An IM client has the ability to enable each of these countermeasures, and specify the amplitude of obfuscation to make her desired tradeoff between performance and resilience. *IMProxy does not require any cooperation from IM providers*, and can be used to obfuscate *any IM service*.

Components of IMProxy: Figure 18 shows the design of IMProxy. For a client to use IMProxy, she needs to install a LocalIMProxy software. LocalIMProxy runs a SOCKS5 proxy listening on a local port. The client will need to change the setting of her IM software (e.g., Telegram software) to use this local port for proxying.

A second component of IMProxy is RemoteIMProxy, which is a SOCKS5 proxy residing outside of the surveillance area. The client needs to enter the (IP,port) information of this remote proxy in the settings of her LocalIMProxy software. Note that, in practice, RemoteIMProxy can be either run by the client herself (e.g., as an AWS instance), or can be run by the IM provider or trusted entities (similar to the MTProto proxies run for Telegram users [56]).

How IMProxy works: Once an IM client sets up her system to use IMProxy as above, her IM traffic to/from the IM servers will be proxied by the proxies of IMProxy, as shown in Figure 18. The IM traffic of the client will be handled by LocalIMProxy and RemoteIMProxy, which obfuscate traffic through padding and delaying.

[4]https://github.com/SPIN-UMass/IMProxy

As shown in the figure, IMProxy acts differently on upstream and downstream IM traffic. For upstream SIM communications (e.g., messages sent by an admin), LocalIMProxy adds padding to the traffic by injecting dummy packets and events at certain locations. First, some dummy packets are injected close to the events in order to change their sizes. The size of padding for each event is chosen randomly, following a uniform distribution in $[0,r_{padding}]$, where $r_{padding}$ is a parameter adjusted by each user. Second, some dummy events (burst of packets) are injected during the silence intervals; this is done randomly: during each 1 second silence interval, an event is injected with a probability $p_{padding}$, where $p_{padding}$ is also adjusted by each individual user. The size of dummy events is drawn from the empirical distribution of the sizes of image messages, as presented earlier. Finally, the dummy packets are removed by RemoteIMProxy before getting forwarded to the IM server. Note that all traffic between LocalIMProxy and RemoteIMProxy is encrypted so the adversary can not identify the dummy packets.

For downstream SIM communications (e.g., messages received by a member), RemoteIMProxy adds dummy packets, as above, which are dropped by LocalIMProxy before being released to the client's IM software. In addition to padding, RemoteIMProxy delays the packets in the downstream traffic. In our implementation, RemoteIMProxy uses an Exponential Distribution with rate λ to generate random delays (which is based on our delay model in Figure 5). Note that no delay is applied on upstream traffic, as the delay will transit to the corresponding downstream traffic.

Note that each client can control the intensity of padding by adjusting the $p_{padding}$ and $r_{padding}$ parameters, and control control the amplitude of delays by adjusting λ.

Implementation: We have implemented IMProxy in Python using the *socketserver* module. We use a Threading TCP Server and Stream Request Handler to implement the SOCKS5 proxy in python. We have released our software as open source.

Evaluation against oblivious adversary: We first evaluate our IMProxy implementation against an adversary who is not aware of how IMProxy works (or its existence). To do so, we evaluate IMProxy against our event-based detector.

Figure 16 shows the ROC curve of the event-based detector for different values of λ. Note that $\frac{1}{\lambda}$ defines the average amount of delay added to the packets. As we can see, *increasing the added delay (by reducing λ) reduces the performance of our attack*, as it causes to missalign events across the monitored flows. For instance, a $\frac{1}{\lambda} = 1s$ reduces the adversary's TP from 93% to 86% (for a constant 10^{-3} false positive).

Figure 17 shows the ROC curves of the event-based detector with different $r_{padding}$ and $p_{padding} = 10^{-4}$. Note that a $p_{padding} = 10^{-4}$ causes a 7% average traffic overhead (please refer to Appendix C on how bandwidth overhead is calculated). As expected, *increasing $r_{padding}$ reduces the performance of our attack*; even a $r_{padding}$ as small of 10% and 7% of dummy events can have a noticeable impact on countering the traffic analysis attacks, i.e., for a 10^{-3} false positive rate,

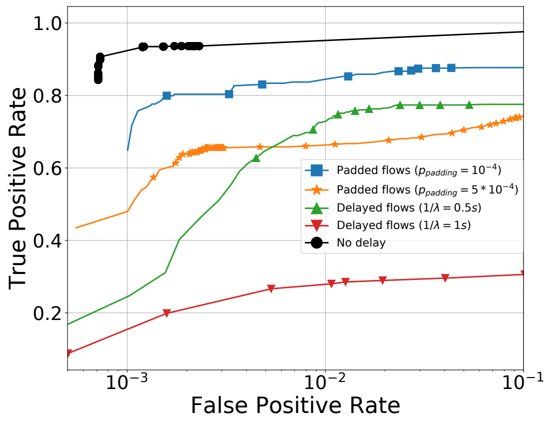


Fig. 19: Evaluating IMProxy against an IMProxy-aware classifier (trained using DeepCorr).

the detection accuracy is reduced from 93% to 62%. Increasing $r_{padding}$ to 50% will further reduce detection accuracy to 56%.

Evaluation against IMProxy-aware adversary: Next, we evaluate IMProxy against an adversary who is aware that target users are deploying IMProxy and also knows the details of IMProxy. Our adversary trains a DeepCorr-based classifier on IM traffic obfuscated using IMProxy (note that our statistical detectors will suffer for such an adversary due to the randomness of IMProxy's obfuscation).

Figure 19 shows the performance of this DeepCorr-based classifier against IMProxy-obfuscated connections (each flow is 15 mins). We use $r_{padding} = 0.1$ and evaluate the performance for different values of $p_{padding}$. As can be seen, *IMProxy is highly effective even against an IMProxy-aware classifier*, demonstrating IMProxy's efficiency in manipulating IM traffic patterns. For instance, for a false positive rate of 10^{-3}, the IMProxy-aware classifier provides true positive rates of 25% and 15% (for average obfuscation delays of 0.5 and 1), which is significantly weaker compared to 93% of the event-based detector when IMProxy is not deployed. As we can see, delaying provides better protection than padding; however, we expect that most users will prefer padding over delays due to the latency-sensitive nature of IM communications.

Note that each user can choose her desired tradeoff between privacy protection and overhead by adjusting the countermeasure parameters. Ideally, the countermeasure software can ask the user her tolerable padding/delay overhead (or her target FP/FN for the adversary), and then will choose the best countermeasure parameters for the user. For instance, based on Figure 19, assuming that a real-world adversary can tolerate a FP of 10^{-3}, if the user states that she intends to keep the adversary's TP below 0.3, the countermeasure software will delay packets with an average of 1s.

VIII. Conclusions

In this paper, we showed how popular IM applications leak sensitive information about their clients to adversaries who merely monitor encrypted traffic. Specifically, we devised traffic analysis attacks that enable an adversary to identify the administrators and members of target IM channels with practically high accuracies. We demonstrated the practicality of our attacks through extensive experiments on real-world IM systems. We believe that our study presents a significant, real-world threat to the users of such services given the escalating attempts by oppressive governments in cracking down on social media.

We also investigated the use of standard countermeasures against our attacks and demonstrated their practical feasibility at the cost of communication overhead and increased IM latency. We designed and implemented an open-source, publicly available countermeasure system, IMProxy, which works for major IM services with no need to support from the IM providers. While IMProxy may be used as an ad hoc, short-term countermeasure by IM users, we believe that to achieve the best usability and user adoption, effective countermeasures should be deployed by IM providers (i.e., through integrating traffic obfuscation techniques into their software). We hope that our study will urge IM providers to take action.

Acknowledgements

We would like to thank Thomas Schneider and Christian Weinert for shepherding our paper, and anonymous reviewers for their constructive feedback. The work has been supported by the NSF grant CNS-1564067 and by DARPA and NIWC under contract N66001-15-C-4067. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

References

[1] "Continued Arrest of Telegram Channels Admin in Orumiyeh," http://kurdistanhumanrights.net/en/continued-arrest-of-telegram-channels-admin-in-orumiyeh/, 2018.

[2] "Admins of 12 Reformist Telegram Channels Arrested in Iran Ahead of May 2017 Election," https://www.iranhumanrights.org/2017/03/12-reformist-telegram-channel-admins-arrested/, 2017.

[3] P. K. Aggarwal, P. Grover, and L. Ahuja, "Security Aspect in Instant Mobile Messaging Applications," in *RAETCS*, 2018.

[4] D. Agrawal, D. Kesdogan, and S. Penz, "Probabilistic treatment of MIXes to hamper traffic analysis," in *IEEE S&P*, 2003.

[5] R. Arratia and L. Gordon, "Tutorial on large deviations for the binomial distribution," *Bull. Math. Biol.*, 1989.

[6] J. Atkinson, M. Rio, J. Mitchell, and G. Matich, "Your WiFi Is Leaking: Ignoring Encryption, Using Histograms to Remotely Detect Skype Traffic," in *MILCOM*, 2014.

[7] A. Back, U. Moller, and A. Stiglic, "Traffic Analysis Attacks and Trade-Offs in Anonymity Providing Systems," in *Information Hiding*, 2001.

[8] A. Balducci and J. Meredith, "Olm cryptogrpahic review." NCC Group PLC, Tech. Rep., 2016.

[9] "At least 60 percent of Iran Internet bandwidth usage accounts for Telegram," https://www.isna.ir/news/96062715757, 2017.

[10] D. Barradas, N. Santos, and L. Rodrigues, "Effective detection of multimedia protocol tunneling using machine learning," in *USENIX Security*, 2018.

[11] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway, "UMAC: Fast and secure message authentication," in *Crypto*, 1999.

[12] J. Black and P. Rogaway, "CBC MACs for arbitrary-length messages: The three-key constructions," in *Crypto*, 2000.

[13] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, "Revealing skype traffic: when randomness plays with you," in *SIGCOMM CCR*, 2007.

[14] N. Borisov, I. Goldberg, and E. Brewer, "Off-the-record Communication, or, Why Not to Use PGP," in *WPES*, 2004.

[15] "Secure P2P Messenger Releases First Version, Receives New Funding," https://briarproject.org/news/2018-1.0-released-new-funding.html, 2018.

[16] X. Cai, R. Nithyanand, and R. Johnson, "CS-BuFLO: A Congestion Sensitive Website Fingerprinting Defense," in *WPES*, 2014.

[17] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, "A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses," in *CSS*, 2014.

[18] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a Distance: Website Fingerprinting Attacks and Defenses," in *CCS*, 2012.

[19] Y.-C. Chang, K.-T. Chen, C.-C. Wu, and C.-L. Lei, "Inferring speech activity from encrypted Skype traffic," in *GLOBECOM*, 2008.

[20] S. Chen, R. Wang, X. Wang, and K. Zhang, "Side-channel leaks in web applications: A reality today, a challenge tomorrow," in *IEEE S&P*, 2010.

[21] T. Chothia and A. Guha, "A statistical test for information leaks using continuous mutual information," in *CSF*, 2011.

[22] C. Cimpanu, "Hong Kong protesters warn of Telegram feature that can disclose their identities," https://www.zdnet.com/article/hong-kong-protesters-warn-of-telegram-feature-that-can-disclose-their-identities/, 2019.

[23] S. E. Coull and K. P. Dyer, "Traffic Analysis of Encrypted Messaging Services: Apple iMessage and Beyond," *SIGCOMM CCR*, 2014.

[24] G. Danezis, "Statistical disclosure attacks," in *IFIP SEC*, 2003.

[25] ——, "The traffic analysis of continuous-time mixes," in *PETS*, 2004.

[26] G. Danezis and A. Serjantov, "Statistical disclosure or intersection attacks on anonymity systems," in *Information Hiding*, 2004.

[27] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-generation Onion Router," in *USENIX Security*, 2004.

[28] D. L. Donoho, A. G. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford, "Multiscale Stepping-Stone Detection: Detecting Pairs of Jittered Interactive Streams by Exploiting Maximum Tolerable Delay," in *RAID*, 2002.

[29] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail," in *IEEE S&P*, 2012.

[30] J. Engler and C. Marie, "Secure messaging for normal people," NCC Group, Tech. Rep., 2015. [Online]. Available: https://www.nccgroup.trust/globalassets/our-research/us/whitepapers/secure-messaging-for-normal-people-whitepaper.pdf

[31] K. Ermoshina, F. Musiani, and H. Halpin, "End-to-End Encrypted Messaging Protocols: An Overview," in *INSCI*, 2016.

[32] O. Flisback, "Stalking anyone on Telegram," https://oflisback.github.io/telegram-stalking/, 2015.

[33] J. Gu, J. Wang, Z. Yu, and K. Shen, "Walls have ears: Traffic-based side-channel attack in video streaming," in *INFOCOM*, 2018.

[34] S. Harris, *CISSP All-in-One Exam Guide*, 6th ed. McGraw-Hill Osborne Media, 2012.

[35] J. Hayes and G. Danezis, "k-fingerprinting: A robust scalable website fingerprinting technique," in *USENIX Security*, 2016.

[36] G. He, M. Yang, X. Gu, J. Luo, and Y. Ma, "A novel active website fingerprinting attack against Tor anonymous system," in *CSCWD*, 2014.

[37] T. He and L. Tong, "Detecting Encrypted Stepping-Stone Connections," *TSP*, 2007.

[38] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naive-bayes classifier," in *CCSW*, 2009.

[39] A. Houmansadr and N. Borisov, "The need for flow fingerprints to link correlated network flows," in *PETS*, 2013.

[40] A. Houmansadr, N. Kiyavash, and N. Borisov, "RAINBOW: A Robust And Invisible Non-Blind Watermark for Network Flows," in *NDSS*, 2009.

[41] ——, "Non-blind watermarking of network flows," *IEEE TON*, 2014.

[42] "Number of mobile phone messaging app users worldwide from 2016 to 2021," https://www.statista.com/statistics/483255/number-of-mobile-messaging-users-worldwide, 2018.

[43] "Getting around Iran's Telegram ban," https://observers.france24.com/en/20180502-getting-around-iran'-telegram-ban-"i-installed-vpn-old-lady-next-door", 2018.

[44] C. Johansen, A. Mujaj, H. Arshad, and J. Noll, "Comparing Implementations of Secure Messaging Protocols (long version)," 2017.

[45] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A critical evaluation of website fingerprinting attacks," in *CCS*, 2014.

[46] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an Efficient Website Fingerprinting Defense," in *ESORICS*, 2016.

[47] D. Kales, C. Rechberger, T. Schneider, M. Senker, and C. Weinert, "Mobile private contact discovery at scale," in *USENIX Security*, 2019.

[48] D. Kedogan, D. Agrawal, and S. Penz, "Limits of anonymity in open environments," in *Information Hiding*, 2002.

[49] "Information leak from chat group. How do we find out which user is sharing information? ," https://security.stackexchange.com/questions/178435/information-leak-from-chat-group-how-do-we-find-out-which-user-is-sharing\-infor, 2018.

[50] B. N. Levine, M. K. Reiter, C. Wang, and M. Wright, "Timing attacks in low-latency mix systems," in *FC*, 2004.

[51] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, and W. Jia, "A new cell-counting-based attack against Tor," *IEEE TON*, 2012.

[52] L. Lu, E.-C. Chang, and M. C. Chan, "Website fingerprinting and identification using ordered feature sequences," in *ESORICS*, 2010.

[53] X. Luo, P. Zhou, E. W. W. Chan, W. Lee, R. K. C. Chang, and R. Perdisci, "HTTPOS: Sealing information leaks with browser-side obfuscation of encrypted flows," in *NDSS*, 2011.

[54] H. M. Moghaddam, B. Li, M. Derakhshani, and I. Goldberg, "Skype-Morph: Protocol Obfuscation for Tor Bridges," in *CCS*, 2012.

[55] "MTProto Mobile Protocol," https://core.telegram.org/mtproto.

[56] "MTProto proxy server for Telegram," https://mtproto.co/.

[57] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *IEEE S&P*, 2005.

[58] M. Nasr, A. Bahramali, and A. Houmansadr, "DeepCorr: Strong Flow Correlation Attacks on Tor Using Deep Learning," in *CCS*, 2018.

[59] M. Nasr, A. Houmansadr, and A. Mazumdar, "Compressive Traffic Analysis: A New Paradigm for Scalable Traffic Analysis," in *CCS*, 2017.

[60] L. H. Newman, "ENCRYPTED MESSAGING ISN'T MAGIC," https://www.wired.com/story/encrypted-messaging-isnt-magic/, 2018.

[61] "200,000,000 Monthly Active Users," https://telegram.org/blog/200-million, 2018.

[62] "[tor-project] Turning on timing obfuscation (iat-mode=1) for some default bridges," https://lists.torproject.org/pipermail/tor-project/2016-November/000776.html.

[63] J.-X. Pan and K.-T. Fang, *Maximum Likelihood Estimation*. New York, NY: Springer New York, 2002, pp. 77–158. [Online]. Available: https://doi.org/10.1007/978-0-387-21812-0_3

[64] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel, "Website Fingerprinting at Internet Scale," in *NDSS*, 2016.

[65] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *WPES*, 2011.

[66] ——, "Website Fingerprinting in Onion Routing Based Anonymization Networks," in *WPES*, 2011.

[67] K. Park and H. Kim, "Encryption is Not Enough: Inferring User Activities on KakaoTalk with Traffic Analysis," in *WISA*, 2015.

[68] H. V. Poor, *An introduction to signal detection and estimation*. Springer Science & Business Media, 2013.

[69] Y. J. Pyun, Y. H. Park, X. Wang, D. S. Reeves, and P. Ning, "Tracing traffic through intermediate hosts that repacketize flows," in *INFOCOM*, 2007.

[70] D. Ramsbrock, X. Wang, and X. Jiang, "A first step towards live botmaster traceback," in *RAID*, 2008.

[71] V. Rimmer, D. Preuveneers, M. Juarez, T. V. Goethem, and W. Joosen, "Automated Website Fingerprinting through Deep Learning," in *NDSS*, 2018.

[72] D. Robertson, "The Licensing and Compliance Lab interviews Guillaume Roguez, Ring Project Director," *Free Software Foundation*, 2016, https://www.fsf.org/blogs/licensing/the-licensing-and-compliance-lab-interviews-guillaume-roguez-ring-project-director.

[73] "Russia orders Telegram to hand over users' encryption keys," https://www.theverge.com/2018/3/20/17142482/russia-orders-telegram-hand-over-user-encryption-keys, 2018.

[74] H. Saribeykan and A. Margvelashvili, "Security Analysis of Telegram," https://courses.csail.mit.edu/6.857/2017/project/19.pdf, 2017.

[75] J. Schectman, "Exclusive: Messaging app Telegram moves to protect identity of Hong Kong protesters," https://www.reuters.com/article/us-hongkong-telegram-exclusive/exclusive-messaging-app-telegram-moves-to-protect-identity-of-hong-kong-protesters-idUSKCN1VK2NI, 2019.

[76] M. Schliep, I. Kariniemi, and N. Hopper, "Is Bob Sending Mixed Signals?" in *WPES*, 2017.

[77] R. Schuster, V. Shmatikov, and E. Tromer, "Beauty and the Burst: Remote Identification of Encrypted Video Streams," in *USENIX Security*, 2017.

[78] V. Shmatikov and M.-H. Wang, "Timing analysis in low-latency mix networks: Attacks and defenses," in *ESORICS*, 2006.

[79] "Most popular mobile messaging apps worldwide as of January 2018, based on number of monthly active users," https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps, 2018.

[80] T. Simonite, "FireChat Could Be the First in a Wave of Mesh Networking Apps," *MIT Technology Review*, 2014. [Online]. Available: https://www.technologyreview.com/s/525921/the-latest-chat-app-for-iphone-needs-no-internet-connection/

[81] Y. Sun, A. Edmundson, L. Vanbever, O. Li, J. Rexford, M. Chiang, and P. Mittal, "RAPTOR: routing attacks on privacy in tor," in *USENIX Security*, 2015.

[82] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *TIFS*, 2017.

[83] "What is Telegram? What do I do here?" https://telegram.org/faq, 2013.

[84] "Where is Telegram based?" https://telegram.org/faq, 2013.

[85] "Telegram and Instagram being restricted in Iran," https://techcrunch.com/2018/01/02/telegram-and-instagram-being-restricted-in-iran, 2018.

[86] "Tor: Pluggable Transports," https://www.torproject.org/docs/pluggable-transports.html.en.

[87] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith, "SoK: Secure Messaging," in *IEEE S&P*, 2015.

[88] "Virtual Private Networking: An Overview," Microsoft Technet, Tech. Rep., 2011. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/bb742566(v=technet.10)

[89] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective Attacks and Provable Defenses for Website Fingerprinting," in *USENIX Security*, 2014.

[90] T. Wang and I. Goldberg, "Walkie-Talkie: An Efficient Defense Against Passive Website Fingerprinting Attacks," in *USENIX Security*, 2017.

[91] X. Wang, S. Chen, and S. Jajodia, "Tracking Anonymous Peer-to-peer VoIP Calls on the Internet," in *CCS*, 2005.

[92] ——, "Network flow watermarking attack on low-latency anonymous communication systems," in *IEEE S&P*, 2007.

[93] X. Wang, D. S. Reeves, and S. F. Wu, "Inter-packet delay based correlation for tracing encrypted connections through stepping stones," in *ESORICS*, 2002.

[94] "WhatsApp reveals major security flaw that could let hackers access phones," https://www.cnn.com/2019/05/14/tech/whatsapp-attack/index.html, 2019.

[95] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson, "Uncovering Spoken Phrases in Encrypted Voice over IP Conversations," *TISSEC*, 2010.

[96] C. V. Wright, S. E. Coull, and F. Monrose, "Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis," in *NDSS*, 2009.

[97] W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao, "DSSS-based flow marking technique for invisible traceback," in *IEEE S&P*, 2007.

[98] Y. Zhang and V. Paxson, "Detecting Stepping Stones." in *USENIX Security*, 2000.

[99] Y. Zhu and R. Bettati, "Unmixing Mix Traffic," in *PETS*, 2005.

[100] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "On flow correlation attacks and countermeasures in mix networks," in *PETS*, 2004.

Appendix

A. *Countermeasures Against Shape-Based Detection*

Figure 20 shows the performance of the shape-based detector while tunneling traffic through circumvention systems. Experiment setting for Tor, VPN, and background traffic has exactly the same settings as in Section VII-A. However, results show that circumvention systems are more impactful on the shape-based detector. Again, only passing traffic through a VPN or Tor does not effect the performance of the detector significantly, and shape-based detector has about an 80% true positive rate while false positive rate is 0.003. Similar to Section VII-A, adding web browsing background traffic or using Tor pluggable transports can impact the performance of the shape-based detection.

B. *Simulations Using Synthetic Traffic*

As mentioned earlier, major SIM services limit the number of channels a client can join; this limits the reliability of our in-the-wild evaluations. To address, we generate synthetic SIM channels to evaluate our detectors on a much larger number of synthetically created IM channels.

Creating Synthetic SIM Communications: We use Algorithm 1 discussed in Section IV-D to create synthetic SIM events (i.e., messages). To convert these SIM events into SIM traffic, we need to simulate the impact of network conditions and other perturbations. Specifically, we apply the effect of network latency according to a Laplacian distribution (see section IV-C). Suppose that $t^{(C)} = \{t_1^{(C)}, \ldots, t_n^{(C)}\}$ and $s^{(C)} = \{s_1^{(C)}, \ldots, s_n^{(C)}\}$ are the vectors of timings and sizes of messages in a synthetic channel, and let $d = \{d_1, \ldots, d_n\}$ be the vector of latencies generated according to a Laplacian distribution. We derive the vector of timings and sizes of the target user flow as follows:

$$\begin{cases} t_i^{(U)} = t_i^{(C)} - d_i, 1 \leq i \leq n \\ s_i^{(U)} = s_i^{(C)}, 1 \leq i \leq n \end{cases}$$

Note that we assume that sizes remain the same in transit. We also simulate the impact of burst extraction noise. Assume the user's bandwidth is bw. Let $t_i^{(l)} = \frac{s_i^{(U)}}{bw}$ be the time it takes the

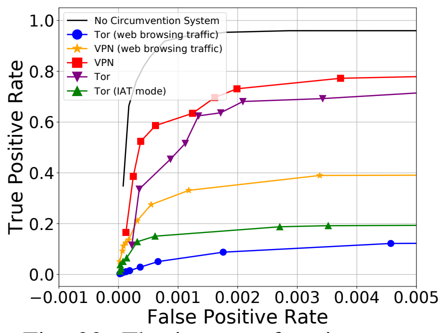


Fig. 20: The impact of various countermeasures on the performance of the shape-based detector (15 minutes of observed traffic).

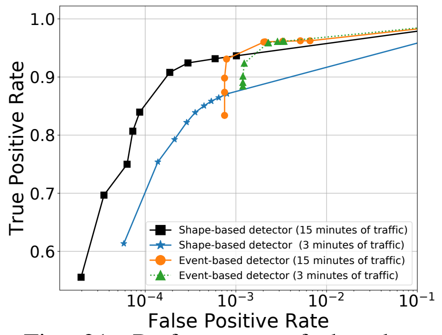


Fig. 21: Performance of the shape-based and event-based detectors over 10,000 synthetic IM connections (15 minutes of observed traffic).

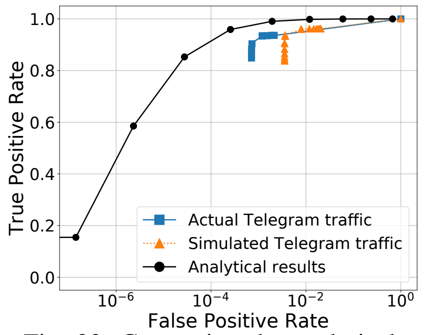


Fig. 22: Comparing the analytical upper bounds of the event-based detector (Section V-A) with empirical results (for 15 minutes of traffic).

target user to send the ith message. We merge two consecutive messages in the target user's traffic if $t_{i+1}^{(U)} - t_i^{(U)} - t_i^{(l)} < t_e$.

Comparing our Two Attacks: We apply our shape-based and event-based detectors on the synthetically-generated SIM traffic. Specifically, we create $10,000$ synthetic SIM communications (in contrast to 500 in the real-world experiments). The channels are divided into five rate buckets. To evaluate false positives, we cross-correlate every SIM client with all the $10,000 - 1$ connects in her rate bucket. Figure 21 shows the ROC curve of the shape-based and event-based detectors over synthetic channels. We can see that, similar to the in-the-wild experiments, the shape-based detector achieves a higher accuracy for smaller false positive rates.

Comparing To Analytical Bounds We also compare our empirical results with the analytical upper bounds of Section V-A. Figure 22 shows the ROC curve of analytical results and experiments using 15 minutes of traffic. As expected, for a fixed false positive rate, the analytical results (formula 1 and 3) upper bound the real-world true positive rate of our event-based detector for both actual and simulated Telegram traffic. Our analytical bound is particularly more useful for smaller values of FP: performing credible experiments for small FPs require a significantly large number of intercepted IM connections which is impractical to capture in experiments.

C. Overhead of Padding Through Dummy Events

TABLE V: Bandwidth overheads of padding through dummy events for different values of $p_{padding}$.

$p_{padding}$	**Bandwidth Overhead**
0.0001	7%
0.0005	34%
0.001	67%

Here, we calculate the bandwidth overhead of adding dummy events as introduced in Section VII-B. We assume that the number of periods in which there is a dummy event follows a Binomial distribution with parameters $p_{padding}$ and the size of the observed flow. Therefore, the average number of dummy events in each flow will follow the mean of the Binomial distribution, which is equal to $p_{padding} \times \ell$, where ℓ is the length of observed flows in seconds. The size of each dummy event is sampled from our collection of IM image messages, which has an average size of $90Kb$. Therefore, we can evaluate the bandwidth overhead of dummy events by dividing the average volume of dummy events over the volume of actual IM messages in long traffic observations. This is shown in Table C for different values of $p_{padding}$.

D. Rate-Based Transition Matrices

The following is the empirical transition probability matrix of the Markov model we use to model IM message sizes (in Section IV-C) for the aggregation of all channels:

$$P = \begin{bmatrix} 0.40 & 0.47 & 0.10 & 0.01 & 0.02 \\ 0.29 & 0.53 & 0.11 & 0.02 & 0.05 \\ 0.19 & 0.36 & 0.40 & 0.02 & 0.03 \\ 0.17 & 0.59 & 0.13 & 0.09 & 0.02 \\ 0.14 & 0.40 & 0.10 & 0.01 & 0.35 \end{bmatrix}$$

The following are the transition matrices for groups of channels with different average daily message rates of 2.31, 7.68, 18.34, 39.47, and 130.57, respectively. We see that the models change slightly for different types of channels.

$$P_1 = \begin{bmatrix} 0.48 & 0.41 & 0.07 & 0.00 & 0.04 \\ 0.28 & 0.52 & 0.11 & 0.01 & 0.08 \\ 0.12 & 0.32 & 0.49 & 0.00 & 0.07 \\ 0.14 & 0.43 & 0.14 & 0.00 & 0.29 \\ 0.13 & 0.44 & 0.06 & 0.00 & 0.38 \end{bmatrix} P_2 = \begin{bmatrix} 0.55 & 0.28 & 0.10 & 0.02 & 0.05 \\ 0.18 & 0.59 & 0.11 & 0.01 & 0.12 \\ 0.13 & 0.35 & 0.45 & 0.01 & 0.07 \\ 0.17 & 0.36 & 0.14 & 0.33 & 0.00 \\ 0.19 & 0.34 & 0.10 & 0.03 & 0.33 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 0.45 & 0.38 & 0.12 & 0.02 & 0.04 \\ 0.22 & 0.55 & 0.13 & 0.03 & 0.07 \\ 0.15 & 0.35 & 0.42 & 0.04 & 0.05 \\ 0.15 & 0.54 & 0.20 & 0.51 & 0.06 \\ 0.13 & 0.31 & 0.10 & 0.03 & 0.43 \end{bmatrix} P_4 = \begin{bmatrix} 0.38 & 0.44 & 0.14 & 0.02 & 0.02 \\ 0.24 & 0.50 & 0.15 & 0.03 & 0.09 \\ 0.17 & 0.35 & 0.43 & 0.03 & 0.03 \\ 0.20 & 0.55 & 0.15 & 0.09 & 0.01 \\ 0.09 & 0.46 & 0.11 & 0.01 & 0.33 \end{bmatrix}$$

$$P_5 = \begin{bmatrix} 0.40 & 0.48 & 0.09 & 0.01 & 0.01 \\ 0.32 & 0.53 & 0.10 & 0.02 & 0.04 \\ 0.21 & 0.37 & 0.39 & 0.02 & 0.02 \\ 0.16 & 0.63 & 0.11 & 0.08 & 0.01 \\ 0.16 & 0.41 & 0.10 & 0.01 & 0.32 \end{bmatrix}$$