A Resilient Randomization Technique Against ML-Driven Timing Side Channel Attack on Mobile Location

Abdeslam El-Yahyaoui[1] and Mohammed Erradi[2]
[1]*CID Development, Rabat, Morocco*
[2]*ENSIAS, Mohammed V University of Rabat, Morocco*

Keywords: User Privacy, Mobile User Location, Timing Side Channel Attack, Random Delays.

Abstract: Delivery status notifications are a standard feature of mobile instant messaging applications. They inform users about the successful delivery of their sent messages. However, this common feature opens up a timing side channel attack compromising user location privacy. This attack exploits variations in Round Trip Times ($RTTs$) across locations, allowing the training of machine learning models for location inference. Recent work proposed a solution based on randomly delaying the $RTTs$ (RDR) on the messenger server side using uniformly sampled perturbations between 0 and a maximum value.
I this work, we have shown that the timing side channel attack still persists with significant accuracy even with the aforementioned randomly delaying RTT countermeasure. We then propose a resilient client side randomization technique involving a distribution with randomly varying parameters across $RTTs$ ($RVPR$). We have shown that the suggested approach ($RVPR$) is resilient against this attack and has less impact on user experience than the existing RDR approach.

1 INTRODUCTION

Mobile messaging applications like WhatsApp have become the primary mode of mobile communications (Schnitzler et al., 2023). They are widely used in: sending and receiving messages, exchanging audio and video calls, informal communication among working colleagues (Loch, 2019), to social interactions among elderly people (Miller et al., 2021). In some cases, messengers are also used for official correspondence with government entities (Purz, 2020), thus composing large and heterogeneous sets of contacts in one application per user. One notable feature of these messaging applications is the real-time status updates provided after delivery and reception of messages. Users can easily track the progress of their messages through the checkmarks received after delivery and read receipt (Ariano, 2020). While this functionality enhances user experience, it also introduces potential privacy issues. As highlighted in (Schnitzler et al., 2023), recipient geolocation could be compromised through this tracking mechanism. This issue is critical because it is simple, rather unsuspicious, and hard to mitigate. Users lack effective means to prevent messages from individuals in their contact list, other than permanently blocking them.

A similar vulnerability was revealed in Short Message Services (SMS) (Bitsikas et al., 2023). Despite the prevalence of smartphones and various messaging apps, $SMSs$ remain an essential communication channel for sending and receiving text messages. They are widely used in diverse applications such as appointment reminders, two-factor authentication, and identity verification(Peeters et al., 2022; Reaves et al., 2016; Reaves et al., 2019). However, as demonstrated in (Bitsikas et al., 2023), $SMSs$ can be exploited to infer a receiver location. The attack works by leveraging SMS Delivery Reports, which are transmitted back to the sender when the network delivers the SMS to the recipient. The sender can request these reports, and there is no way for the recipient to prevent them (the recipient cannot prevent the sender from requesting and getting the Delivery Reports of the sent messages). By measuring $RTTs$ between sending an SMS and receiving the corresponding Delivery Report, the attacker can distinguish various locations of the target recipient and determine their location area after a training phase. Imagine the scenario of a person (the victim) conducting a press conference from a particular location, such as their official residence. Despite public knowledge of the victim's current location and

El-Yahyaoui, A. and Erradi, M.
A Resilient Randomization Technique Against ML-Driven Timing Side Channel Attack on Mobile Location.
DOI: 10.5220/0013457900003979
In *Proceedings of the 22nd International Conference on Security and Cryptography (SECRYPT 2025)*, pages 61-71
ISBN: 978-989-758-760-3; ISSN: 2184-7711

phone number, the adversary initiates the collection of round-trip time measurements by sending silent or regular *SMS* messages to the victim. These measurements create timing signatures specific to the location. Later, when the attacker seeks to determine if the victim has returned to their residence, they simply send a silent *SMS* and compare the timing signatures. The attacker can always contact the victim and collect the round-trip time measurements at any time with just their mobile phone number. This is possible because Delivery Reports are inherent to protocol specifications across all mobile network generations and cannot be disabled.

An existing solution (Schnitzler et al., 2023; Bitsikas et al., 2023) consists of obfuscating the $RTTs$ by adding random delays sampled from a uniform distribution on the messenger server side or within the core network for *SMS*. This is not sufficient to avoid this timing side channel attack and adversely affects server and core network resources.

In this paper, we propose a more resilient client side distribution where the parameters exhibit random variation across RTT measurements. We will show that this randomization technique renders the computation of the $RTTs$ more challenging. The main contributions of this work are:

1. Proofing the persistence of the timing side channel attack even with the existing approach.
2. The design of a randomization technique resilient against location inference attack.
3. Proofing the usefulness of implementing the solution on the client side rather than the server side to avoid its high resource consumption.
4. The experiments of the suggested solution outperform the existing one.

The subsequent sections of the paper are organized as follows: Section 2 discusses related works, followed by preliminaries in Section 3. Section 4 presents the threat model. Section 5 elaborates on the proposed solution. Section 6 provides the experimental results. Finally, Section 7 concludes the paper.

2 RELATED WORK

The most related works to our research are those conducted in (Schnitzler et al., 2023) and (Bitsikas et al., 2023). The former focused on messenger applications and demonstrated a location inference attack based on the $RTTs$ generated by delivery status notifications. They proposed various solutions, including randomly delaying these notifications by a uniformly sampled random delay between 0 and 5 seconds. The latter demonstrated the existence of a similar side-channel attack in *SMS* services. This attack uses $RTTs$ associated with delivery reports received post-sending an *SMS*. They also suggested different solutions, including randomly delaying the delivery reports. In our study, we propose a novel approach more resilient against location inference attacks.

Several research works have extensively explored the relationship between timing and distance, integrating the Internet's topology to enhance localization techniques (Candela et al., 2019; Du et al., 2020; Katz-Bassett et al., 2006; Kohls and Diaz, 2022). These studies collectively contribute to advancing the understanding and application of timing-based localization and distance estimation on the Internet. Other studies (Hong et al., 2018; Hussain et al., 2019; Lakshmanan et al., 2021) worked on localizing cellular network users either actively or passively through the capture of identifiers. Kotuliak et al. (Kotuliak et al., 2022) showcased enhanced localization accuracy of up to 20 meters using Timing Advance and leveraging overshadowing techniques (Erni et al., 2021; Yang et al., 2019).

Authors in (Michalevsky et al., 2015) investigated a side channel vulnerability allowing applications to access aggregate power usage data. By analyzing short-term power consumption, these applications could infer the user location, since the phone location affects the power consumption of its cellular radio. Oberhuber et al. (Oberhuber et al., 2025) conducted another study on sensor-based side-channel attacks targeting Android smartphones. Their research specifically explored the exploitation of power-related signals through the Android sensor framework. The findings demonstrated how commonly available sensors, such as the geomagnetic rotation vector sensor, can be used to leak sensitive information in realistic attack scenarios.

The security threats posed by in-app advertisements on mobile devices were studied in (Diamantaris et al., 2021). The study highlighted the ability of in-app ads to access mobile sensors without specific permissions. This critical issue exposes sensitive user data like the geographical location. Han et al. (Han et al., 2022) studied a location spoofing attack targeting Wi-Fi positioning systems. The attack involves crafting a Location Lookup Table containing estimated Wi-Fi access point locations based on cell towers and Wi-Fi nodes. This attack enables adversaries to infer users' daily activities and monitor their movements. Researchers in (Pourali et al., 2022) investigated existing privacy analysis tools for mobile apps, noting their focus on standard HTTP and HTTPS

channels while potentially overlooking vulnerabilities in non-standard ones like TCP/UDP and custom encryption. Their research revealed widespread GPS data leakage in mobile apps.

Several studies have explored side-channel attacks in cryptography. Nassi et al. (Nassi et al., 2023) introduced optical cryptanalysis, a novel side-channel attack where secret keys are extracted by measuring light emitted from a device's power *LED* using a photodiode and analyzing subtle fluctuations during cryptographic operations. Erata et al. (Erata et al., 2023) introduced Pascal, a tool designed to detect power side-channel vulnerabilities in constant-time cryptographic code using symbolic register analysis techniques. Ahmed et al. (Ahmed et al., 2024) provided a review of deep learning use for enhancing encryption techniques against side-channel attacks between 2018 and 2024. They discussed the vulnerabilities in cryptographic algorithm implementations, particularly through power analysis and explores how deep learning methods such as Convolutional Neural Networks and Recurrent Neural Networks can be employed to mitigate these threats.

In their work, Zhang et al. (Zhang et al., 2024) examined micro architectural timing side-channel attacks and their countermeasures. They focused on vulnerabilities arising from performance optimizations like simultaneous multi-threading and out-of-order execution. Maar et al. (Maar et al., 2025) introduced *KernelSnitch*, a novel side-channel attack based on timing differences in kernel data structures to leak sensitive information. The data structures, such as hash tables and trees, exhibit timing variances depending on their occupancy levels, which can be exploited by an unprivileged attacker to infer information about kernel operations or activities in other user processes.

Cristiano et al. (Rodrigues et al., 2024) introduced a novel class of side-channel attacks on microcontrollers. These attacks exploit subtle timing differences in the bus interconnect arbitration logic. The authors presented *BUSted*, a method that exploits these timing differences to bypass memory protection mechanisms and leak sensitive data from isolated memory regions. Bognar et al. (Bognar et al., 2023) proposed enhancing an instruction set architecture with micro architectural leakage traces obtained through profiling. This approach facilitates the development of tools to detect and mitigate side-channel vulnerabilities.

3 PRELIMINARIES

Round Trip Time (RTT) is the time in milliseconds (ms) it takes for a network packet to go from a starting point to a destination and back to the starting point. Authors in (Schnitzler et al., 2023) and (Bitsikas et al., 2023) demonstrated the existence of a timing side channel vulnerability in mobile messaging applications and $SMSs$ based on $RTTs$. In this section, we will provide some background on side channels as well as the concept of the considered side channel attack.

3.1 Side Channels and User Privacy

On a computing system, the occurrence of any event will inevitably produce side effects which could be observed by the party not supposed to know anything about the event (Wang et al., 2023). These side effects are channels from which protected information can be inferred, which can compromise users privacy. User privacy refers to an individual's right to control how their personal data is used (Bateman, 2023). It should be collected and used according to their choices, and in a way that helps them remain anonymous. According to the General Data Protection Regulation, the European Union regulation on information privacy, personal data covers users location (Bateman, 2023). A user could be profiled based on the places that he visits, albeit at a low frequency. He might not be willing to reveal that he is visiting a particular church, a health clinic, or some hotel. Revealing these places might enable an entity with access to the user location to profile him (Fawaz and Shin, 2014).

3.2 Messaging Applications RTT-Based Side Channel Attack

In the context of messaging applications (Schnitzler et al., 2023), the involved $RTTs$ are derived from the delivery status notifications shown in Figure 1. The sent message initially reaches a messenger server, which then acknowledges receipt by sending a delivery notification to the sender. The server forwards the message to the ultimate receiver, subsequently receiving confirmation of successful reception. Upon receiving this notification, the server dispatches a confirmation of successful delivery back to the sender. This process is simplified since sender and receiver can be connected to different messenger servers.

Based on these exchanges, the adversary calculates $RTT(S,M)$ between **S**ender and **M**essenger server as the time difference between sending the message and receiving the first notification from the

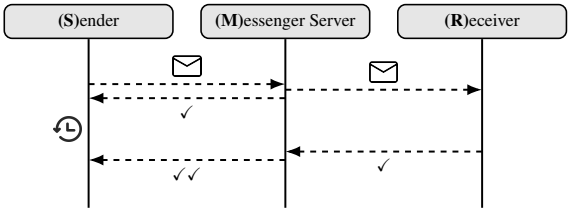

Figure 1: Message flow scenario.

server. Likewise, $RTT(S,R)$ between **S**ender and **R**eceiver as the duration between sending the message and receiving the second notification confirming successful delivery to the intended receiver. From the two obtained $RTTs$, the adversary calculates the $RTT(M,R)$ between **M**essenger and **R**eceiver as follows:

$$RTT(M,R) = RTT(S,R) - RTT(S,M) \quad (1)$$

For a specific location A, the calculated RTT can be expressed as $\tau(A) + \varepsilon$. Here, ε accounts for the measurement's fluctuation around a constant value $\tau(A)$, influenced by factors such as varying network and messenger server loads. This is why the authors opted for a sequence of 5 RTT measurements to construct their timing dataset, as this improves accuracy in accounting for these variations. After training a model on this dataset, the adversary can predict locations based on newly measured $RTTs$.

The variability in Round-Trip Times ($RTTs$) is not solely affected by distance; rather, it is influenced by a multitude of factors. For instance, fluctuations in network traffic affect how long it takes to send messages and receive delivery confirmations. Additionally, the specific network path traversed by packets can differ based on the Internet service providers involved, leading to distinct timing characteristics for each location. Consequently, locations exhibit different timing signatures, consisting of the distribution of $RTT(M,R)$ considering these diverse factors. Notably, even locations equidistant from the server might have different timing patterns.

3.3 SMS RTT-Based Side Channel Attack

A similar study on the Short Message Service (*SMS*) (Bitsikas et al., 2023) demonstrated that delivery reports of sent *SMSs* lead also to a timing side-channel vulnerability. Receiving an *SMS* inevitably generates Delivery Reports whose reception bestows a timing attack vector at the sender. The high-level idea of this attack is as follows: The time elapsed between sending an *SMS* and receiving the corresponding Delivery Report differs depending on the receiver location. Therefore, one can distinguish different receiver locations by observing the elapsed time.

The attack is based on *SMS* Delivery Reports, which are transmitted back to the sender when the network delivers the *SMS* to the recipient. The sender can request these reports, and there is no way for the recipient to prevent them. By measuring the round-trip time, i. e., the time elapsed between sending an *SMS* and receiving the corresponding Delivery Report, the attacker can distinguish various locations of the target recipient and determine their location area after a training phase.

As depicted in Figure 2, *SMS* transmissions prompt acknowledgment notifications from the core network and Delivery Reports from the recipient. This Delivery Report indicates the *SMS* delivery status. To calculate the fingerprint for location identification, the attacker uses three timestamps: the *SMS* transmit time (t_{txt}) when the attacker sends the *SMS*, the *SMS* sent time (t_{sent}) when the attacker receives the "*ACK*" notification, and the *SMS* delivery time (t_{del}) when the attacker receives the Delivery Report notification. From these timestamps, the real sent duration (T_{sent}), real delivery duration (T_{del}), total delivery duration (T_{tot}), and delivery ratio (P) are computed. These features are calculated for each individual *SMS* transmission.

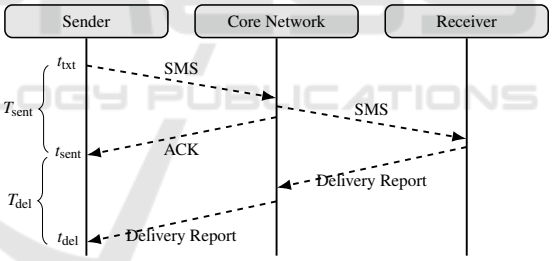

Figure 2: SMS timing features.

$$T_{sent} = t_{sent} - t_{txt} \quad (2)$$

$$T_{del} = t_{del} - t_{sent} \quad (3)$$

$$T_{tot} = T_{del} + T_{sent} \quad (4)$$

$$P = \frac{T_{del}}{T_{tot}} = \frac{t_{del} - t_{sent}}{t_{del} - t_{txt}} \quad (5)$$

Additionally, to generate robust location signatures, the attacker considers two consecutive *SMS* transmissions ($i-1$ and i) and estimates the differences in real sent duration ($T_{\Delta\text{sent}}$) and real delivery duration ($T_{\Delta\text{del}}$).

$$T_{\Delta\text{sent}} = (T^{i}_{sent} - T^{i-1}_{sent})/T^{i-1}_{sent} \quad (6)$$

$$T_{\Delta\text{del}} = (T^{i}{}_{del} - T^{i-1}_{del})/T^{i-1}_{del} \quad (7)$$

The location signature comprises the six features: (T_{sent}, T_{del}, T_{tot}, P, $T_{\Delta\text{sent}}$, $T_{\Delta\text{del}}$).

4 THREAT MODEL

The attacker's goal is to locate the victim receiver's whereabouts, specifically, whether the victim's mobile is in a specific geographic area of interest. We have two different scenarios to perform the considered attack; using either messenger applications or *SMSs*:

Messenger Application Scenario:

- Both the attacker and the victim have regular devices supporting the considered messaging application.
- The attacker can access and analyze his own network traffic to extract timing information.
- Adversary and victim must be in each other's contact lists in the messenger (the threat is limited to parties who likely know each other).
- The involved devices have network connection.
- The adversary does not require physical access to mobile devices, or any network entities.

SMS Scenario:

- The considered *SMSs* can be regular private *SMSs*, marketing *SMSs*, or silent *SMSs*.
- The adversary knows the victim's mobile number and can send it *SMSs*.
- The adversary can target any valid mobile number attached to a cellular provider.
- The adversary does not require physical access to mobile devices, or any network entities.

In the next section, we will elaborate on our suggested solution.

5 A RANDOMLY VARYING PARAMETERS DISTRIBUTION APPROACH

Our solution is based on enhancing the existing approach that consists of randomizing the $RTTs$ using a standard uniform distribution. In this section, we highlight this approach weaknesses, then we present our suggested one.

5.1 Existing Approach

The existing approach consists of delaying the $RTTs$ using samples from the uniform distribution $\mathcal{U}(a,b)$ having a and b as parameters. The added delays are implemented on the messenger server for messaging applications and core network for *SMSs*. We will show that even with this solution, the attack can still be performed. Additionally, we will examine the impact of implementing this solution on the messenger server or the core network side.

Persistence of the Attack. The considered distribution exhibits static properties such as min, max, and mean values corresponding to a, b, and $(b-a)/2$, respectively. Statistically estimating these features is easy based on a sequence of a relatively small number n of samples. For example, if an adversary is interested in a location A with an RTT of $\tau(A)+\varepsilon$, a sequence of n measurements at this location will produce values like $\tau(A)+\varepsilon+u_1$, $\tau(A)+\varepsilon+u_2$, ..., $\tau(A)+\varepsilon+u_n$. u_1, u_2, ..., u_n are sampled from $\mathcal{U}(a,b)$ and ε is the the fluctuation of the RTT around $\tau(A)$. The adversary can select the minimum value of this sequence, which is $\tau(A)+\varepsilon+min(u_1,u_2,...u_n)$, approximately $\tau(A)+\varepsilon+a$. Consequently, they can deduce $\tau(A)+\varepsilon$ if a is known. They repeat this process five times to construct a sequence of five approximated $RTTs$ that the model requires to predict the location. Algorithm 1 shows this attack procedure when a is known. The same process applies to the mean and max values. If a and b are unknown, the adversary can't directly deduce the $RTTs$. Thus they will need to construct a shifted dataset based on $RTT+a$, $RTT+b$, or $RTT+(b-a)/2$ to carry out the attack. We will further discuss the effectiveness of the attack in such scenarios in section 6. It's important to note that performing the attack based on min, max, or mean values makes the dataset construction more complex since it requires more measurements.

Impact on the Messenger Server. Implementing the solution on the messenger server side would degrade its resources since it is designated to handle a large number of messages. The same constraint applies to the core network for *SMSs*. Thus, we will focus our reasoning only on the messaging applications scenario.

The dynamics of the server can be modeled like a system with an inflow and an outflow. The balance of inflow and outflow determines the state of the server.

We consider a server with:

- *Capacity* (K): Maximum number of messages the server can hold.
- *Incoming Speed* (s_i): Rate at which messages arrive at the server.

Algorithm 1: Min value based attack process for a given location A.

Input: Model `M` trained to classify a set of locations including a location A
Input: The lower bound a of the uniform distribution $\mathcal{U}(a,b)$ used to randomize the RTTs
Input: A list `RTT_sequences` of five sequences of n randomized RTTs at location A using $\mathcal{U}(a,b)$
Output: Prediction of the location

```
Initialize an empty list estimated_RTTs;
for i ← 1 to 5 do
    RTT_sequence ← RTT_sequences[i];
    min_sequence ← min(RTT_sequence);
    estimated_RTT ← (min_sequence − a);
    Append estimated_RTT to
     estimated_RTTs;
prediction ← M(estimated_RTTs) ;
 // Use M to predict location
return prediction;
```

- *Outgoing Speed* (s_o): Rate at which messages are processed.

The server is saturated when it reaches its maximum capacity and cannot accept new messages until some are processed.

Saturation Condition. For the sake of simplicity, we consider (s_i) and (s_o) constants. Saturation occurs when the incoming rate exceeds the outgoing rate. In such scenario, the net flow rate into the server is given by:

$$NetFlowRate = s_i - s_o \tag{8}$$

The Formula for Saturation Time: The number of messages in the server at time t is approximately given by:

$$n(t) = n_0 + (s_i - s_o) \cdot t \tag{9}$$

where n_0 is the initial number of messages in the server. This formula shows a linear growth of messages over time.

The server reaches its capacity K at t_{sat}:

$$K = n_0 + (s_i - s_o) \cdot t_{sat} \tag{10}$$

For $s_i > s_o$, the time until saturation t_{sat} is given by:

$$t_{sat} = \frac{K - n_0}{s_i - s_o} \tag{11}$$

This formula is applicable under the condition that $s_i > s_o$ and $n_0 \leq K$. If $s_i \leq s_o$, the server will not reach saturation as the outflow is sufficient to manage or exceed the inflow. Implementing the solution on the server side would significantly reduce s_o, as each message handling duration will be delayed by $(b-a)/2$ in average. For $a = 0$ and $b = 5$ as suggested in (Schnitzler et al., 2023), the messages will be delayed by 2.5 seconds in average. During peak hours, when the server receives a high volume of messages, s_i would exceed s_o, leading to server saturation.

5.2 The Suggested Solution

We propose using a client side uniform distribution with parameters randomly varying across RTT measurements. These parameters are sampled from uniform distributions at each RTT measurement. Specifically, this client side distribution has the form: $\mathcal{U}(u(a_1,b_1),u(a_2,b_2))$, with a_1, b_1, a_2 and b_2 appropriately chosen. The distribution $\mathcal{U}(u(a_1,b_1),u(a_2,b_2))$ is characterized by its difficulty in capturing its minimum, maximum, and mean values. This makes it more difficult for attackers to remove randomness by repeatedly sending messages. **We considered only uniform distributions since they offer the highest entropy (MacKay, 2003), thus, a significant level of randomness in the sampled values.**

Implementing the solution on the client side wouldn't cause significant issues on the client device. In fact, a typical client device is expected to handle a very limited number of messages. However, in this case, we recommend implementing an anti-Denial of Service (anti-DOS) mechanism to protect the device. This is crucial, as an attacker could flood the device that implements the random delays with a high volume of messages, causing resource exhaustion. This scenario is similar to server saturation under heavy load conditions and the application of random delays (as demonstrated in Section 5.1).

We compare the behaviors of the two distributions $\mathcal{U}(0,5)$ suggested in (Schnitzler et al., 2023) and $\mathcal{U}(u(0,2),u(3,5))$ having the same minimal, maximal, and mean values. Figure 3 shows the values of 100 samples from both distributions. Notably, we observe that our proposed distribution seldom reaches values close to its real minimum and maximum compared to the other one. For deeper insights Figures 4, 5, and 6 depict the minimum, maximum, and mean values of varying size samples from both distributions. Each round of measurements involves sampling a new set of values, without accumulation with the previous ones. These plots show the large difference between the required sample size to obtain values near the min, max, and mean of the two dis-

tribution. For instance, Figure 4 shows that $\mathcal{U}(0,5)$ attained values near 0 for samples of size almost 10, whereas $\mathcal{U}(u(0,2),u(3,5))$ necessitated nearly a 230-size sample.

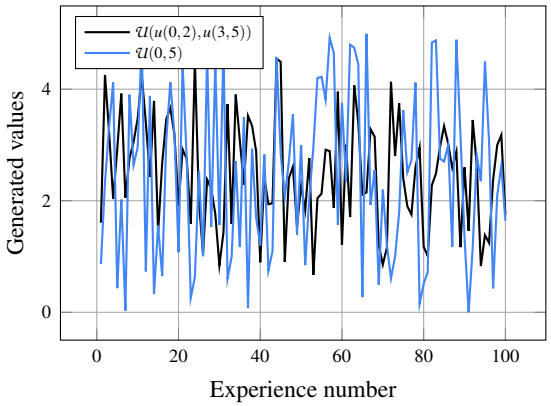

Figure 3: Behavior of the two distributions.

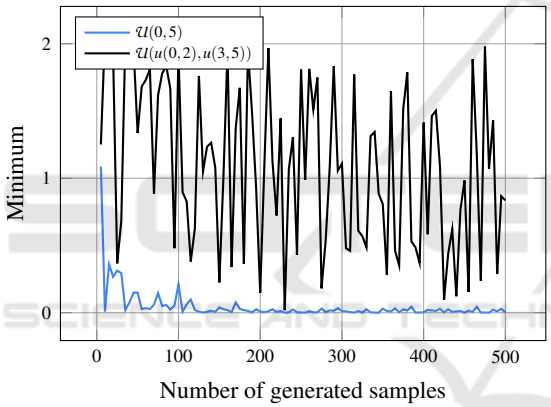

Figure 4: Min values comparison.

Table 1 shows the number of obtained values in the window $[0,0.1]$ for different sample sizes. For the distribution $\mathcal{U}(u(0,2),u(3,5))$, we didn't obtain any value between 0 and 0.1 for sample sizes less than 400. We then observed one value in the interval for the sizes 600 and 1000 and two values with the sample size 5000. For the distribution $\mathcal{U}(0,5)$, we obtained 1 value between 0 and 0.1 for sample sizes 10 and 50. This number increased significantly along the increasing sample sizes until reaching 217 for a sample size of 10000. In summary, as the sample size increases, the number of values near 0 also increases for both distributions, but the rate of increase is much higher for the distribution $\mathcal{U}(0,5)$ compared to $\mathcal{U}(u(0,2),u(3,5))$. This is expected as the support of $\mathcal{U}(0,5)$ covers the interval [0, 5] uniformly, whereas $\mathcal{U}(u(0,2),u(3,5))$ is constrained by the distributions $\mathcal{U}(0,2)$ and $\mathcal{U}(3,5)$.

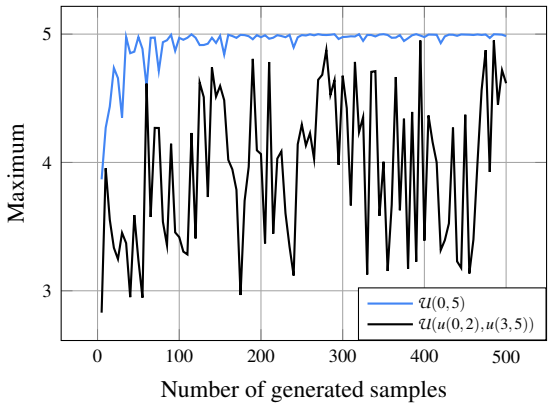

Figure 5: Max values comparison.

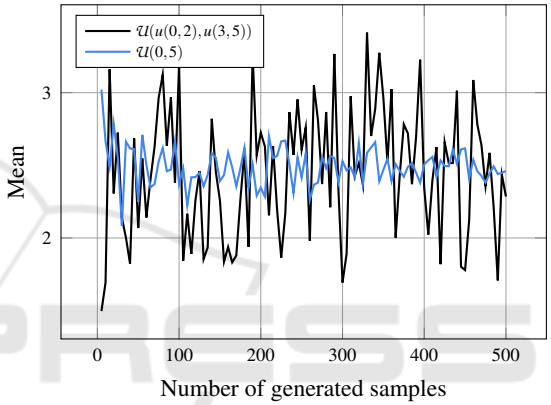

Figure 6: Mean values comparison.

Impact on User Experience: Figure 3 presenting the behavior of the two distributions shows that the distribution, $\mathcal{U}(0,5)$, exhibits greater fluctuations between its min and max values compared to $\mathcal{U}(u(0,2),u(3,5))$. This entails that for the first one, a sender may receive message notifications either promptly or significantly delayed, leading to potential drawbacks in user experience. However, in the second scenario, notifications of message reception are less time dispersed, offering a more consistent experience for the sender.

In the next section, we will conduct experimental validation of the theoretical results.

6 EVALUATION

In this section we aim to experimentally evaluate the effectiveness of our approach and compare it to the existing one.

Table 1: Number of values between zero and 0.1 for different sample sizes.

Distribution \ Sample size	10	50	100	200	300	400	600	1000	2000	5000	10000
$\mathcal{U}(u(0,2),u(3,5))$	0	0	0	0	0	0	1	1	2	2	7
$\mathcal{U}(0,5)$	1	1	2	5	5	7	13	22	33	92	217

6.1 Dataset Construction

For the dataset collection, we calculated sequences of *RTTs* based on ping replay timings between a sender located in Morocco and two distant servers. The two servers are respectively located in Rabat, Morocco and California, spanning a significant distance of almost $9,645km$. We focused exclusively on these two locations to construct a dataset serving as a proof of concept for the attack. We conducted measurements at three different moments of the day: at 10:00, 12:00, and 16:30. During these measurements, we alternated between Ethernet and cellular connections. Similar to (Schnitzler et al., 2023), we accounted for sequences of 5 *RTT* measurements. The dataset comprises 15,386 rows in total.

6.2 Minimum Value Based Attack

In our initial dataset, there are no random delays, resulting in an accuracy of over 97% when training an *LSTM* model on it. From this dataset, we used a subset containing 400 homogeneous data points for testing purposes. We simulated the implementation of the two randomization techniques and evaluated the resulting accuracy of the attack.

For each *RTT* in the testing dataset, we sampled a sequence of delays from $\mathcal{U}(u(0,2),u(3,5))$ (respectively $\mathcal{U}(0,5)$), and added their minimum to the *RTT*. This process mimics an adversary attacking the solutions using the min value. Subsequently, we trained the *LSTM* model on the non randomized dataset portion and evaluated its accuracy on the randomized portion. We conducted 20 rounds of training and prediction for different sample sizes and represented the mean accuracy along with the standard deviation in Figure 7.

For $\mathcal{U}(0,5)$, increasing the sample size significantly enhances the attack accuracy. This is because, with a larger number of samples, the probability of obtaining minimum values closer to the real *RTTs* becomes higher. Essentially, with more data points, the likelihood of accurately estimating the *RTTs*, and thus inferring the correct locations, increases. In scenarios where there are no restrictions on the number of messages that can be sent, such as with silent *SMSs*, the attack can achieve an accuracy exceeding 90%. However, the situation is totally different when

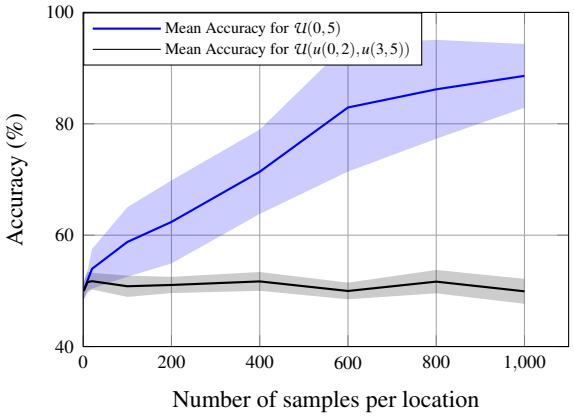

Figure 7: Mean accuracy of minimum-based attack.

considering the distribution $\mathcal{U}(u(0,2),u(3,5))$. Despite increasing the sample size, the attack accuracy remains close to 50%. Consequently, the attacker is essentially making a binary decision, choosing between two potential locations with an accuracy being near 50%. This indicates that the attack is no better than random guessing.

6.3 Case of Unknown Distribution Parameters

Previously, we demonstrated the difficulty in accurately capturing the min, max, and mean values of $\mathcal{U}(u(0,2),u(3,5))$. Consequently, the accuracy of attacks relying on these values is notably low, as exemplified by the attack based on the minimum value. Thus, conducting such an attack with this distribution is unfeasible, especially when the parameters are unknown beforehand. In this subsection, we will focus on evaluating the feasibility of the attack when the parameters of $\mathcal{U}(a,b)$ are unknonwn. As mentionned before, the attacker can construct a dataset based on $RTT + cte$, where cte is a, b, or $(b-a)/2$.

Initially, we shifted the dataset *RTTs* by the same constant, and computed the accuracy for 20 rounds of model training. We varied the shifting constant value and plotted the mean accuracy along with its standard deviation in Figure 8. The experimental results indicate a notable decrease in accuracy as the shifting constant increases demonstrating that even slight variations in dataset shifting can degrade performance.

Particularly, the accuracy approaches approximately 50% when the shifting constant exceeds 1s (1000ms).

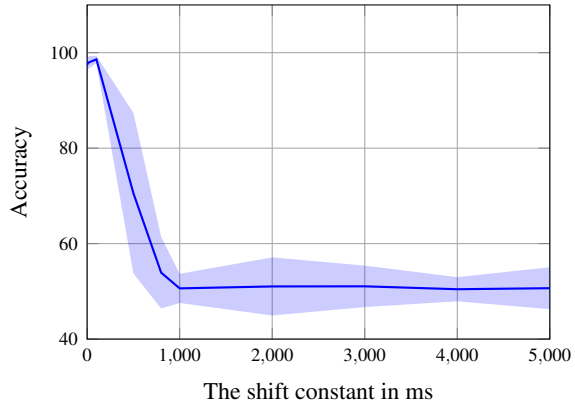

Figure 8: Shift effect on accuracy.

We then normalized and standardized the shifted dataset and re-evaluated the resulting accuracy. The outcomes in Figure 9 reveal that this improves accuracy, stabilizing at approximately 64% for larger shifting constants. This indicates that the attack is also feasible in this scenario. However, the accuracy is lower than conducting the attack with known parameters.

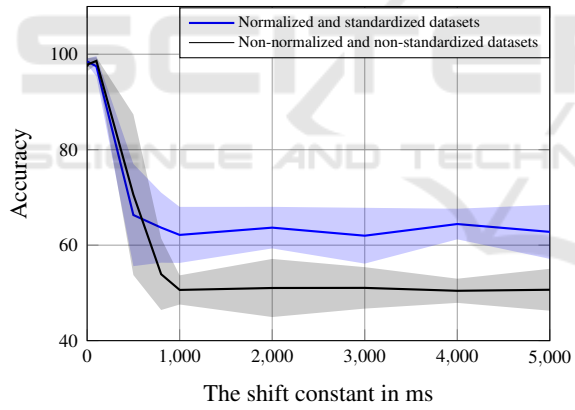

Figure 9: Normalization and standardization.

6.4 Impact on the Messenger Server

As we theoretically demonstrated in Section 5.1, implementing the solution on the server side would lead to a degradation of server resources. To test this, we created a server with a queue and simulated 1000 clients sending messages to this server. The clients send messages and wait for a random duration between messages to mimic user behavior. This waiting duration is sampled from a *Poisson* distribution with the parameter 5. This indicates an average of 5 seconds between two consecutive messages. The sent messages are placed into the server queue. We considered two configurations: one where the server introduces a random delay sampled from $\mathcal{U}(0,5)$, and one without these delays. We recorded the evolution of the server queue size over one minute and plotted the results in Figure 10. In the normal case,

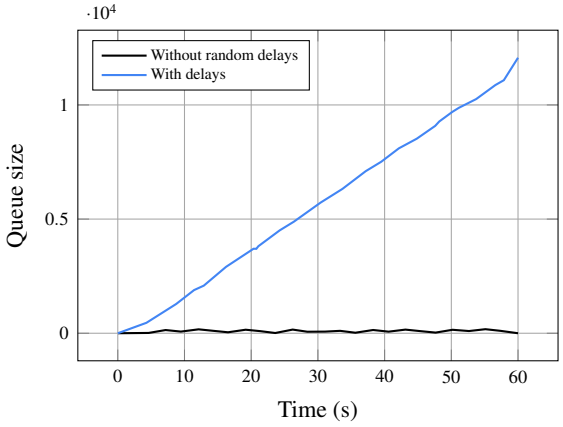

Figure 10: Server queue size variation over time.

the server queue size fluctuates around zero, indicating efficient handling of incoming messages without overhead. This means that the server processes messages at a rate that is equal to or greater than the arrival rate, preventing any accumulation. However, when random delays are introduced, the situation changes significantly. The queue size increases linearly due to message accumulation. This result is compatible with Equation 9 with $n_0 = 0$ (n_0 is the initial number of messages in the server). In this scenario, the reception rate exceeds the server's processing rate, causing messages to arrive faster than they can be handled, resulting in a continuously growing queue. This observation validates our theoretical findings, demonstrating that the introduction of random delays substantially affects server performance.

7 CONCLUSIONS AND FUTURE WORK

This paper addresses the privacy concerns associated with timing side-channel attacks on mobile messaging applications and *SMS* services. Existing solutions propose using uniform random delays to obfuscate Round Trip Times on the messenger server and *SMS* core network sides. Our evaluation demonstrates the persistence of the attack with significant accuracy with this solution in place and showed the significant impact it has on the messenger server and *SMS* core network resources. We proposed a more resilient client side randomization technique using a distribu-

tion with randomly varying parameters across RTT measurements. Through experimental validation, we highlighted the effectiveness of our proposed solution compared to the existing one.

However, we could not use the original dataset due to privacy restrictions. Our illustrative dataset comprises $RTTs$ derived from ping reply timings to construct a *POC* of the attack. Therefore, the obtained results may vary slightly if we were to use data based on $RTTs$ from messenger notifications or *SMS* delivery reports. Furthermore, our study has limitations in the exploration of machine learning algorithms, as we solely opted for an *LSTM* model to demonstrate the feasibility of the attack. Additionally, we only focused on two locations in our work to construct a Proof of Concept of the attack. As a future work, we aspire to investigate the potential existence of timing side-channel vulnerabilities in video and voice calls.

REFERENCES

Ahmed, A. A., Hasan, M. K., Aman, A. H., Safie, N., Islam, S., Ahmed, F. R. A., and Rzayeva, L. (2024). Review on hybrid deep learning models for enhancing encryption techniques against side channel attacks. *IEEE Access*.

Ariano, R. (2020). 'what do the check marks mean on whatsapp?': How to determine the status of your message on whatsapp. Accessed March 12, 2024.

Bateman, R. (2023). Privacy practices for user location. https://www.termsfeed.com/blog/user-location-privacy-practices/, as of April 2024.

Bitsikas, E., Schnitzler, T., Pöpper, C., and Ranganathan, A. (2023). Freaky leaky sms: Extracting user locations by analyzing sms timings. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 2151–2168.

Bognar, M., Winderix, H., Bulck, J. V., and Piessens, F. (2023). Microprofiler: Principled side-channel mitigation through microarchitectural profiling. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, pages 651–670. IEEE.

Candela, M., Gregori, E., Luconi, V., and Vecchio, A. (2019). Using ripe atlas for geolocating ip infrastructure. *IEEE Access*, 7:48816–48829.

Diamantaris, M., Moustakas, S., Sun, L., Ioannidis, S., and Polakis, J. (2021). This sneaky piggy went to the android ad market: Misusing mobile sensors for stealthy data exfiltration. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1065–1081.

Du, B., Candela, M., Huffaker, B., Snoeren, A. C., and claffy, k. (2020). Ripe ipmap active geolocation: Mechanism and performance evaluation. *ACM SIGCOMM Computer Communication Review*, 50(1):4–10.

Erata, F., Piskac, R., Mateu, V., and Szefer, J. (2023). Towards automated detection of single-trace side-channel vulnerabilities in constant-time cryptographic code. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, pages 687–706. IEEE.

Erni, S., Leu, P., Kotuliak, M., Röschlin, M., and Capkun, S. (2021). Adaptover: Adaptive overshadowing of lte signals. *ArXiv*, abs/2106.05039.

Fawaz, K. and Shin, K. G. (2014). Location privacy protection for smartphone users. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 239–250.

Han, X. et al. (2022). Location heartbleeding: The rise of wi-fi spoofing attack via geolocation api. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

Hong, B., Bae, S., and Kim, Y. (2018). Guti reallocation demystified: Cellular location tracking with changing temporary identifier. In *25th Annual Network and Distributed System Security Symposium (NDSS 2018)*, San Diego, California, USA.

Hussain, S. R., Echeverria, M., Chowdhury, O., Li, N., and Bertino, E. (2019). Privacy attacks to the 4g and 5g cellular paging protocols using side channel information. *Network and distributed systems security (NDSS) symposium2019*.

Katz-Bassett, E., John, J. P., Krishnamurthy, A., Wetherall, D., Anderson, T., and Chawathe, Y. (2006). Towards ip geolocation using delay and topology measurements. In *ACM Internet Measurement Conference (IMC '06)*, pages 71–84, Rio de Janeiro, Brazil.

Kohls, K. and Diaz, C. (2022). Verloc: Verifiable localization in decentralized systems. In *USENIX Security Symposium (USENIX '22)*, Boston, MA, USA.

Kotuliak, M., Erni, S., Leu, P., Röschlin, M., and Capkun, S. (2022). Ltrack: Stealthy tracking of mobile phones in lte. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1291–1306.

Lakshmanan, N., Budhdev, N., Kang, M. S., Chan, M. C., and Han, J. (2021). A stealthy location identification attack exploiting carrier aggregation in cellular networks. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3899–3916.

Loch, P. (2019). Whatsapp in the workplace. Accessed April 2024.

Maar, L., Juffinger, J., Steinbauer, T., Gruss, D., and Mangard, S. (2025). Kernelsnitch: Side-channel attacks on kernel data structures. In *Network and Distributed System Security Symposium 2025: NDSS 2025*.

MacKay, D. J. (2003). *Information theory, inference and learning algorithms*. Cambridge University Press.

Michalevsky, Y. et al. (2015). Powerspy: Location tracking using mobile device power analysis. In *24th USENIX Security Symposium (USENIX Security 15)*.

Miller, D., Abed Rabho, L., Awondo, P., de Vries, M., Duque, M., Garvey, P., Haapio-Kirk, L., Hawkins, C., Otaegui, A., Walton, S., et al. (2021). *The global smartphone: Beyond a youth technology*. UCL Press.

Nassi, B., Vayner, O., Iluz, E., Nassi, D., Jancar, J., Genkin, D., and Elovici, Y. (2023). Optical cryptanalysis: Recovering cryptographic keys from power led light fluctuations. In *Proceedings of the 2023 ACM SIGSAC*

Conference on Computer and Communications Security, pages 268–280.

Oberhuber, M., Unterguggenberger, M., Maar, L., Kogler, A., and Mangard, S. (2025). Power-related side-channel attacks using the android sensor framework. In *Network and Distributed System Security Symposium 2025: NDSS 2025*.

Peeters, C., Patton, C., Munyaka, I. N. S., Olszewski, D., Shrimpton, T., and Traynor, P. (2022). Sms otp security (sos): Hardening sms-based two factor authentication. In *ASIA CCS'22: ACM Asia Conference on Computer and Communications Security*, pages 2–16, Nagasaki, Japan. ACM.

Pourali, S., Samarasinghe, N., and Mannan, M. (2022). Hidden in plain sight: exploring encrypted channels in android apps. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS)*.

Purz, M. (2020). How governments worldwide are using messaging apps in times of covid-19. https://www.messengerpeople.com/governments-worldwide-covid-19/, as of April 2024.

Reaves, B., Scaife, N., Tian, D., Blue, L., Traynor, P., and Butler, K. R. B. (2016). Sending out an sms: Characterizing the security of the sms ecosystem with public gateways. In *IEEE Symposium on Security and Privacy (SP)*, pages 339–356, San Jose, CA, USA. IEEE Computer Society.

Reaves, B., Vargas, L., Scaife, N., Tian, D., Blue, L., Traynor, P., and Butler, K. R. B. (2019). Characterizing the security of the sms ecosystem with public gateways. *ACM Transactions on Privacy and Security*, 22(1):2:1–2:31.

Rodrigues, C., Oliveira, D., and Pinto, S. (2024). Busted!!! microarchitectural side-channel attacks on the mcu bus interconnect. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 3679–3696. IEEE.

Schnitzler, T., Kohls, K., Bitsikas, E., and Pöpper, C. (2023). Hope of delivery: Extracting user locations from mobile instant messengers. In *Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA. The Internet Society.

Wang, Z., Guan, J., Wang, X., Wang, W., Xing, L., and Alharbi, F. (2023). The danger of minimum exposures: Understanding cross-app information leaks on ios through multi-side-channel learning. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 281–295.

Yang, H., Bae, S., Son, M., Kim, H., Kim, S. M., and Kim, Y. (2019). Hiding in plain signal: Physical signal overshadowing attack on lte. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 55–72, Santa Clara, CA. USENIX Association.

Zhang, J., Chen, C., Cui, J., and Li, K. (2024). Timing side-channel attacks and countermeasures in cpu microarchitectures. *ACM Computing Surveys*, 56(7):1–40.