Fingerprinting Smartphones Remotely via ICMP Timestamps

Marius Cristea and Bogdan Groza

***Abstract*—Fingerprinting mobile devices over a WiFi channel has both positive and negative security implications: on one hand it allows the establishment of physically secure identifications by exploiting physically unclonable characteristics, on the other it jeopardizes privacy by mediating remote identification without user awareness. We are able to distinguish between smartphones within minutes, whenever their clock drifts apart with around one part-per-million, by using innocuous ICMP timestamps. To achieve this, we compute the clock skew of the device with linear programming techniques, a previously known methodology. Our experiments are done on some of the top Android devices on the market and the results show that remote identification is feasible even in the presence of: discontinued connections, clock synchronization, multiple hops or poor network conditions. Since blocking ICMP timestamps is not available by default on Android platforms, as well as editing the drivers is not within reach for average users, it seems that this may be indeed a privacy hole for smartphone users (and may be even worse for the closed-source platforms).**

***Index Terms*—802.11, ICMP, smartphone, timestamp.**

I. Introduction and motivation

THE smartphones market has spectacularly grown in the last years, clearly taking over the traditional phones. This is not unexpected due to the numerous advantages that come from clever applications that help users to keep in touch with their friends, e.g., social networks and social apps, or companies to track their customers in order to provide more suitable products.

But all these come at a price, especially on the side of privacy. Notably, some apps track users without their consent and, intentionally or not, it may be impossible to turn the tracking functionalities off. A popular example is the tracking feature of iOS 4 [1]. But even if disabling such features is possible, users can be further deceived in that they are poorly aware of the fact that their devices can be fingerprinted whenever they connect to wireless hot-spots. Even if solutions for enhancing the privacy of mobile users exist, these solutions do not completely solve the problem as shown in [4]. The research here raises a distinct privacy concern that arises from the possibility of identifying a smartphone without user's awareness from ICMP timestamps.

Usually, the identification is performed via the MAC address of the device. But there are countless applications that can be used to forge this code (even available in the Google Play store). However, the MAC address is not the only way in which a device can be identified. In their seminal work, Kohno et al. [3] showed that remote physical device identification from clock skews is feasible even without the cooperation of the user. By applying the same ideas on Android smartphones, we discovered that they can be uniquely identified by exploiting otherwise innocuous ICMP (Internet Control Message Protocol) timestamps whenever the devices connect to some wireless network. This has obvious impact on the privacy of the user since at any WiFi (802.11) hot-spot it may be possible to decide if the same device was or not previously connected.

Related Work. The work in [3] uses clock skews to physically identify devices, it can be applied without the device's cooperation, and it doesn't require any modifications of the applications running on the device. To achieve this, the authors exploit the TCP timestamp option. They also account for ICMP exploitation, but this option is not further analyzed because their setting is over a WAN and ICMP can be blocked by some firewall. Being a passive method it needs a lot of TCP packets for clock skew estimation, requiring about 1 hour for capturing them. There are two significant differences with smartphones. First, all devices that we had contact with have the ICMP timestamp option enabled and very likely there is only a small minority of users that are aware of this. Second, RFC 792 imposes a 1 milliseconds resolution to the ICMP timestamps and, since we use active requests for them, sufficient timestamps can be collected in a short amount of time, which makes the method feasible for fast phone identification (in particular around 15 minutes appear to be sufficient in our experiments for a precision of around 1 ppm). In the context of smartphones, an overview of the non-cryptographic methods available for identification is presented in [6], this also accounts for the possibility of using clock skews but this is not further explored. Several refinements for estimating clock skews are accounted in [2], here we stay with the linear programming method from [5].

II. Methodology

We first describe the main ideas behind computing the clock skew based on ICMP packets, and then we present the experimental results and a brief analysis of their accuracy.

A. ICMP packets

Let $\mathcal{C}$ be the challenger, i.e, the role played by the access point, which computes the clock skew of responder $\mathcal{R}$, i.e., the role played by the smartphones. To fingerprint the smartphone with its clock skew, $\mathcal{C}$ sends an ICMP Timestamp Requests (ICMP type 13 message) which will trigger the mobile device to respond with an ICMP Timestamp Response (ICMP type 14 message). From the fields of an ICMP packet, of interest to us are: *type* which can be 13 for timestamp request or 14

Manuscript received February 24, 2013. The associate editor coordinating the review of this letter and approving it for publication was C. Mitchell.

This work is partially supported by the strategic grant POSDRU/88/1.5/S/50783, 2009-2013.

The authors are with the Faculty of Automatics and Computers, Politehnica University of Timisoara, Romania (e-mail: {marius-simion.cristea, bogdan.groza}@aut.upt.ro).

Digital Object Identifier 10.1109/LCOMM.2013.13.130419

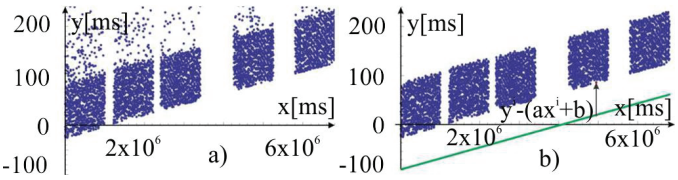

Fig. 1. Unfiltered offset a) and clock skew estimation b)

for response, *originate timestamp* which is the time set by the sender just before sending the message ($\mathcal{T}_\mathcal{C}$), *receive timestamp* which is the time when $\mathcal{R}$ receives the message ($\mathcal{T}_\mathcal{R}$) and *transmit timestamp* is the time value set by $\mathcal{R}$ just before sending the reply. The timestamps from ICMP messages are 32 bits of milliseconds since midnight UT.

For the i^{th} message we take $\mathcal{T}_\mathcal{C}^i$ as the *originate timestamp* and $\mathcal{T}_\mathcal{R}^i$ as the *receive timestamp*. Then $x^i = \mathcal{T}_\mathcal{C}^i - \mathcal{T}_\mathcal{C}^0$ represents the difference in time between the i^{th} message and the first message and $y^i = \mathcal{T}_\mathcal{R}^i - \mathcal{T}_\mathcal{R}^0 - (\mathcal{T}_\mathcal{C}^i - \mathcal{T}_\mathcal{C}^0)$ represents the observed offset. Now, if a smartphone p has received n ICMP timestamp requests, then let $\mathcal{O}_p = \{(x^i, y^i), i = \overline{0,n}\}$ be the offset for the smartphone p, and τ_p the clock skew of p, which will be the slope of the points in $\mathcal{O}_p$. An example is shown in Fig. 1 a). If the smartphone would have no clock skew, then $\mathcal{O}_p$ will be represented as a horizontal line.

B. Computing the clock skew

The clock skew computation is based on the method presented by [5]. For this, we need to identify the line $(ax + b)$ that represents a lower bound for $\mathcal{O}_p$ (see Fig. 1 b)), this means that: $y^i - (ax^i + b) >= 0, i = \overline{0,n}$. This line needs to be as close as possible to $\mathcal{O}_p$, in order to achieve this, the sum of distances between the line and all the points in $\mathcal{O}_p$ must be minimized: $\min\{\sum_{i=0}^{n} y^i - (ax^i + b)\} = \min\{\sum_{i=0}^{n} y^i - a\sum_{i=0}^{n} x^i - nb\}$. By using these relations the problem of finding the clock skew can be seen as a linear programing problem, where the inequality represents the problem's constraints and the second relation its objective function. Note that although noise is present in the measurements, linear programming is not affected by it because it is not present below the offset. For clarity in some of the figures that follow, the noise is filtered.

C. Experimental results

In the experiments we used Wolfram's Mathematica 5.0 for solving the linear programming problem and generating the graphics. The smartphones used in the setup were 5 Android based devices: Samsung Google Nexus S running Android Jelly Bean 4.1.1, Motorola Motoluxe running Android Gingerbread 2.3.6, Samsung Galaxy Mini S5570 running Android Gingerbread 2.3.5 and two Samsung I9000 Galaxy S running Android Gingerbread 2.3.6. Android allows ICMP timestamp requests and responds to them. Besides this, there is no setting to block ICMP timestamp requests and there isn't any app on the Google Play Store to do this. The clock skew of each of the phones, computed via the ICMP timestamps, is shown in Fig. 2. The differences are significant and clear even for the two identical Samsung Galaxy devices (Fig. 2 d). In Table I we report the clock skews computed in ppm's ($\mu s/s$).

TABLE I
CLOCK SKEW FOR THE PHONES USED IN EXPERIMENT

Phone	No. Timestamps (Packets)	Capture time	Clock skew
Nexus S	7000	11.66 min	22.68 ppm
Motorola Motoluxe	5000	8.33 min	87.43 ppm
Galaxy Mini	5000	8.33 min	- 3.17 ppm
Galaxy S (a)	7097	11.82 min	20.83 ppm
Galaxy S (b)	5000	8.33 min	16.26 ppm

D. Quality of the results

The quality of our measurements can be evaluated along three distinct coordinates that we discuss next: accuracy (the degree of closeness to the true value), precision (the reproducibility of the measurements) and resolution (the smallest change that can be detected).

To determine the *accuracy* of the measurements we need a reference value which is the *true* value of the clock skew. In the absence of a more precise measurement, we fixed this reference value as the clock skew that is measured for the phone staying fixed and close to the wireless end-point with no input from the user. Obviously, the data in the experiments has plenty of noise, but by increasing the sample size, the *precision* of the measurement increases as well (this is expected in the presence of systematic errors). To show that this is indeed the case, Fig. 4 a) shows the convergence of the skew with the number of measurements for all the five phones. After several thousand packets (which can be captured in around 15 minutes) the skew modifies only after the decimal point which suggests a precision greater than 1ppm. The number of packets that we capture can be reduced as we are mainly interested in more time for the clock offset to cumulate. We further made several test scenarios to check if the new measurements are similar to the reference value. The clock skew remained stable even if the measurements are stopped then resumed later and the same happened even if NTP (Network Time Protocol) was used for time synchronization. Indeed, in both cases a discontinuity appears in the plot, but this does not change the slope of the line that lower-bounds the values as shown in Fig. 3 a) and b). The clock skew also remained stable even when fingerprinting was done over multiple hops (we tried over three hops in our experiment), if the smartphone user was moving or if the network conditions were bad. More noise is present in the latter case, but the noise appears only above the offset and not below it. These are shown in Fig. 3 c) and d).

The last question is: which is the smallest variation that we can detect, i.e., the *resolution* of our measurements? The results obtained on timestamps that we captured in around 15 minutes suggest that a clock skew of around 1 ppm can be detected. To confirm this, we started a distinct experiment by compromising the clock skew of one of the devices in the experimental setup. We modified the handler of icmp requests from the driver of the WRT routers which is *ip_icmp.c*. For smartphones, the same can be done on older versions of Android, while for more recent ones modifying *icmp.c* inside the kernel is needed. The driver was modified such that to each timestamp of value x it adds an extra $10^{-6}x$

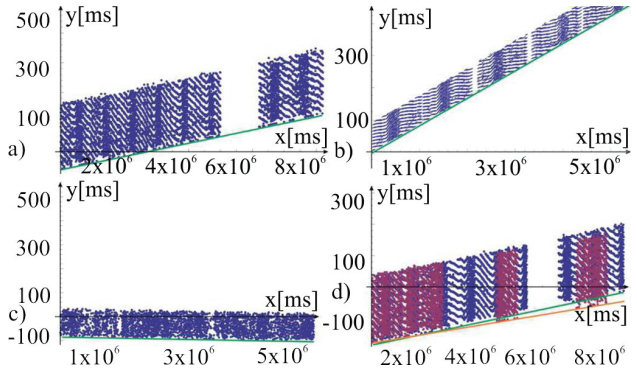

Fig. 2. Clock offsets for: a) Samsung Google Nexus S; b) Motorola Motoluxe; c) Samsung Galaxy Mini S5570; d) two Samsung I9000 Galaxy S

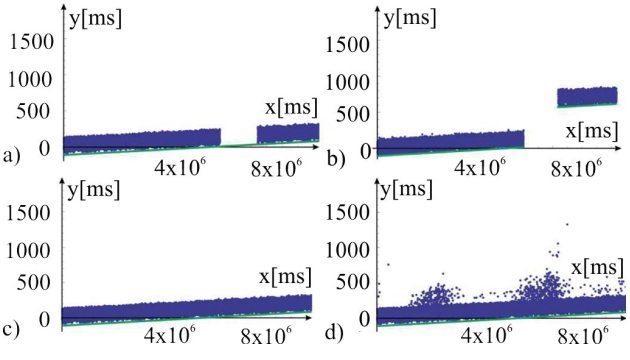

Fig. 3. Clock offset for Samsung Google Nexus S: a) for discontinuous measurements; b) with NTP enabled; c) over three hops d) for bad network conditions (phone moving and several WiFi devices around)

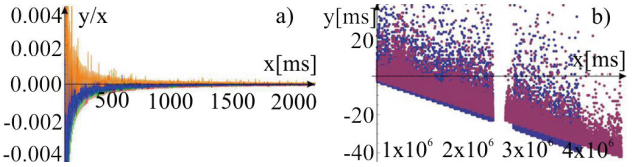

Fig. 4. a) Convergence of the skew relative value (y^i/x^i); b) The original clock skew of a device vs. the skew affected by a drift of 10^{-7}

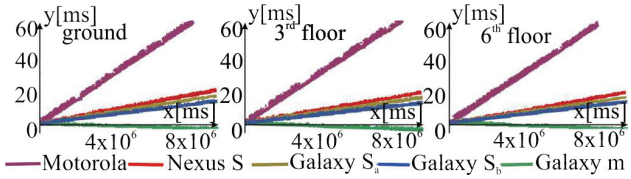

Fig. 5. The clock skew recorded on 3 different access points

(this adds 1 part to each million). We were clearly able to recognize this difference. By lowering this error to $10^{-7}x$ we needed around an hour to see a clear difference, see Fig.4 b). Increasing the fingerprinting time even more would allow us to go well beyond the decimal point, but we believe that this would be rather unrealistic, since a smartphone user is not likely to spend hours connected to some wireless hot-spot. Technically speaking, no two oscillators are identical and the offset between two clocks will gradually increase over time. Thus, given enough time we can distinguish between any two clocks provided that their skew is constant.

In order to test if the traceability scenario is or not realistic, we mounted 3 distinct 802.11 access points on different floors of the university and distributed the 5 experimental smartphones to 5 persons. The participants were required to connect to them in some random order and use the Internet for around 15 minutes (no unusual latency was noted while browsing). The clock skews from the experiment are presented in Fig. 5. The results showed that it's easy to identify the users as well as the order in which they connect to the APs.

III. Conclusion

Clearly, the clock skew can be used to identify smartphones and this may affect the privacy of users, e.g., it may hold as evidence for their presence at a particular hotspot or that they download particular data, etc.

The problem can be alleviated by either disabling ICMP timestamp requests or by changing the slope of the offset. As a Linux based operating system, Android could benefit from the powerful firewall available on Linux: *iptables*. This firewall can be also used to log the ICMP timestamp requests, so that an app can read these logs and notify the user that he might be a subject of fingerprinting. The user can allow the ICMP timestamp requests or block them. Since blocking the timestamps may make the user appear suspicious, another way to make the phone untraceable is by altering the timestamp itself. This is possible by modifying the network driver but more tedious work is needed.

Very likely there is little user awareness on the fact that ICMP timestamps can be exploited to recover their phone identity. The fact that there is no straight-forward method for average users to turn off timestamps makes this more problematic, as the phone can be traced without the user's consent or a possibility to do something against this.

References

[1] A. Allan and P. Warden, "Got an iPhone or 3G iPad? Apple is recording your moves," @o'reilly, Apr 2011. Available: http://radar.oreilly.com/2011/04/apple-location-tracking.html, accessed 01/02/2013.

[2] J. Bi, Q. Wu, and Z. Li, "On estimating clock skew for one-way measurements," *Computer Commun.*, vol. 29, no. 8, pp. 1213–1225, 2006.

[3] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Trans. Dependable Secur. Comput.*, vol. 2, no. 2, pp. 93–108, Apr. 2005.

[4] I. Krontiris, F. C. Freiling, and T. Dimitriou, "Location privacy in urban sensing networks: research challenges and directions," *Wireless Commun.*, vol. 17, no. 5, pp. 30–35, Oct. 2010.

[5] S. B. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," technical report, Amherst, MA, USA, 1998.

[6] K. Zeng, K. Govindan, and P. Mohapatra, "Non-cryptographic authentication and identification in wireless networks," *Wireless Commun.*, vol. 17, no. 5, pp. 56–62, Oct. 2010.