On Reliability of Clock-Skew-Based Remote Computer Identification

Libor Polčák[1] and Barbora Franková[1]

[1] *Faculty of Information Technology, Brno University of Technology, Božetěchova 2, 612 66 Brno, Czech Republic*
ipolcak@fit.vutbr.cz, xfrank08@stud.fit.vutbr.cz

Keywords: Device Fingerprinting, Clock Skew, Security, Counter-measures.

Abstract: Clocks have a small in-built error. As the error is unique, each clock can be identified. This paper explores remote computer identification based on the estimation of clock skew computed from network packets. The previous knowledge of the method is expanded in various ways: (1) we argue about the amount of data that is necessary to get accurate clock skew estimation, (2) the study of different time stamp sources unveils several irregularities that hinders the identification, and (3) the distribution of clock skew in real network makes the precise identification hard or even impossible.

1 INTRODUCTION

Each computer has internal clock to measure time. As the manufacturing process is not precise on atomic level, each clock has its own deficiencies. Consequently, each computer measures time with its own in-built inaccuracy, *clock skew*. When the inaccuracy accumulates, clock skew impact becomes visible and measurable.

In remote clock-skew-based computer identification, a detector (*fingerprinter*) gathers time stamps of computers to be identified (*fingerprintees*) with the goal of unique identification of all computers. (Kohno et al., 2005) proposed tracking ICMP and TCP time stamps. Later, others introduced other sources and studied the method. Previously reported results suggest high reliability and applicability of the clock-skew-based identification.

This paper revisits the results of previous studies of clock-skew-based identification and focuses on the applicability in real networks. This paper contributes in the following areas:

- The duration of a clock skew estimation is more important than the number of evaluated time stamps.
- There is an anomaly in TCP time stamps of Apple devices.
- Clock skew is not unique enough for networks of several hundred computers.
- All users running Windows, Mac OS, Linux and their derivatives, e.g. Android or iOS, can easily evade the identification.

This paper is organized as follows. Section 2 overviews the method to estimate clock skew and discusses previous work. Section 3 elaborates on the data required for sufficient clock skew estimates. Section 4 explores the time stamp properties and lists irregularities observed during experiments. Section 5 evaluates the identification in real network; the results show that the method is not precise enough to uniquely identify devices. The impact of the findings is considered in Section 6 and the method is compared to similar identification methods. Section 7 concludes the paper.

2 CLOCK SKEW ESTIMATION

The original idea behind clock skew computation to identify network devices was introduced by (Kohno et al., 2005). Let us denote the time reported by clock C at time t (as defined by national standards, i.e. the *true time*) as $\mathrm{R}_C(t)$. The offset is a difference between two clocks: $\mathrm{off}_{C,D}(t) = \mathrm{R}_C(t) - \mathrm{R}_D(t)$. Assume that $\mathrm{off}_{C,D}$ is a differentiable function in t, then, *clock skew* $\mathrm{s}_{C,D}$ is the first derivative of $\mathrm{off}_{C,D}$. Clock skew is measured in μs/s, generally denoted as *parts per million* (ppm).

Consider C to be the clock of the fingerprinter and D to be the clock of the fingerprintee. As R_D is not observable by the fingerprinter, it sees packets marked with time stamps delayed by $\varepsilon(t)$, i.e. the network delay. If ε were a constant, the first derivative of $\mathrm{off}'_{C,D} = \mathrm{R}_C(t) - \varepsilon - \mathrm{R}_D(t)$ would have been equal to the first derivative of $\mathrm{off}_{C,D}$. Unfortunately, ε is not a constant.

Let us represent observed packets from the fingerprintee as *offset points* (x,y) where x is the observation time, either $\mathrm{R}_C(t)$ or the elapsed time since the start of the measurement; and y is the observed offset $\mathrm{off}'_{C,D}(t)$ Kohno et al. proposed to estimate clock skew by the slope of the upper bound of all offset points. They have shown that the slope of the upper bound is similar to the slope of the $\mathrm{off}_{C,D}$. Consequently, the first derivatives are similar and the clock skew can be estimated by computing the slope of the upper bound of all offset points. See Figure 1 for an example.

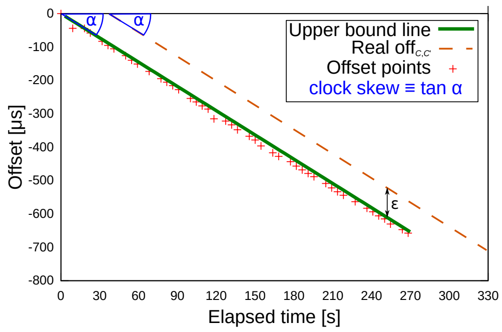

Figure 1: Clock skew estimation.

The original paper studied the advantages and disadvantages of the clock-skew-based identification. (Kohno et al., 2005) used two sources of time stamps: time stamps from TCP can be collected passively whereas gathering time stamps from ICMP needs active approach. The downside of the ICMP-based measuring is the non-uniform implementation in current operating systems described in Section 4. TCP time stamps (Jacobson et al., 1992) are present in each TCP segment header when the client and the server negotiate this option during the initial TCP phase. As Kohno et al. observed, TCP time stamps are not generated by Windows machines by default. We evaluated that this is valid to this date and Windows 8.1 still does not add TCP time stamps by default.

Later, time stamps carried on the application layer were considered (Murdoch, 2006; Zander and Murdoch, 2008; Ding-Jie Huang et al., 2012). Zander and Murdoch computed clock skew from time stamps present in HTTP protocol. Ding-Jie Huang et al. applied AJAX to send additional time stamps to the web server that computes clock skew of its clients as one of the input of multi-factor authentication. In this case, the fingerprintee accessed a special web page prepared by the operator of the fingerprinter.

In addition, separate line of study emerged in the field of rogue access point identification (Jana and Kasera, 2010; Lanze et al., 2012). This approach uses yet another source of time stamps — IEEE 802.11 Time Synchronization Function time stamps exchanged in Wi-Fi networks.

(Sharma et al., 2012) improved the detection by introducing a batch of ICMP packets to compensate for the latency and loses on the network path. We did not consider using this approach for two reasons: (1) our primary focus was on passive detection, (2) ICMP time stamps are not supported by Windows in a standard way, and (3) ICMP time stamps are disabled by default in Apple operating systems.

Although (Ding-Jie Huang et al., 2012) improved the clock skew estimation by linear regression, the reported error was still in the range of ± 1 ppm, i.e. the same as discussed by (Kohno et al., 2005). Hence, we did not use their improvements.

One of our previous work (Polčák et al., 2013) reported that NTP-enabled Linux hosts became immune to the clock skew computed from TCP. This paper elaborates on NTP influence on other sources of time stamps.

3 ACCURACY OF ESTIMATES

This section revisits the discussion (Kohno et al., 2005; Sharma et al., 2012) about the time required to accurately estimate the clock skew of a specific computer. This is important especially for applications that need to decide whether a new short-time measurement (Ding-Jie Huang et al., 2012) matches a previously calculated value. (Sharma et al., 2012) claim that a minimum number of 70 system time stamps are required before the computed clock skew estimation becomes stable. However, our measurements suggest that simply stating the number of required time stamps is not enough.

For example, a computer uploading a large file through a 1 Gbps network link generates up to 81,274 Ethernet frames per second (the longest Ethernet frames takes 1,538 B on the wire including inter frame gap, preamble, and frame check sequence). 70 consecutive packets are sent within about 861 μs. Consider a computer with in-built clock skew of 100 ppm (which is relatively large considering the study presented in the Section 5). The inaccuracy caused by the clock skew after 861 μs is only 86 ns. Current operating systems update the clock with frequency of 10–1000 Hz, i.e. the resolution of the clock value is in milliseconds. Thus, the inaccuracy cannot be visible during this short period of time.

To illustrate that the estimation of clock skew depends on time rather than the number of packets, we captured a communication between two computers to a pcap file. The original captured pcap file was sampled into seven sample files. Each of the sample files contained the closest packets distributed with a gap of at least n seconds. The considered gaps between consecutive packets were 1, 3, 5, 10, 20, 30, and 60 seconds. So the file number 1 contained 60-times more packets than the file number 7. We estimated the clock skew after each received packet.

The data from the whole measurement, which took more than 330 minutes, are shown in the Figure 2. The shape of the curves, formed by the points of estimated clock skew, suggests that the estimated clock skew value of about -30.2 ppm can be improved even after such a long time. However, shortly after the beginning of the experiments, the clock skew estimation was in the range of ± 1 ppm as originally described by (Kohno et al., 2005) and used also by (Ding-Jie Huang et al., 2012).

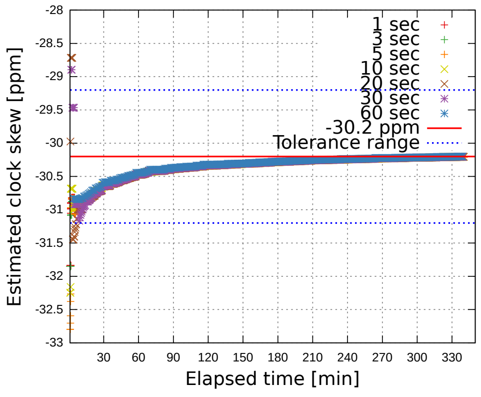

Figure 2: The clock skew estimation can be improved even after several hours of measurement. The duration of the measurement is the dominant factor in clock skew estimation as the seven estimates converge to the final value with a similar pace.

The detail view at the beginning of the clock skew estimation is depicted in Figure 3 and 4. As shown in the Figure 3, estimation of clock skew from all sample sets are in the range of -31.2 ppm to -29.2 ppm after about 2 minutes and the precision of the clock skew computation does not depend on the number of packets. Although Figure 4 contains the four most sparse sample files, the clock skew estimation is in the desired range after 4–6 minutes. Note that the second largest data file, with 20 seconds packet gaps, takes the longest time to fit into the final range. This is because the external influence, such as the state of the network, is heavier than the benefits of having higher number of packets available.

Since (Sharma et al., 2012) claim: *For TCP measurements, the same set of 100 time stamps was yielded in only a few minutes of data capture depending on the network load*. We believe that our findings

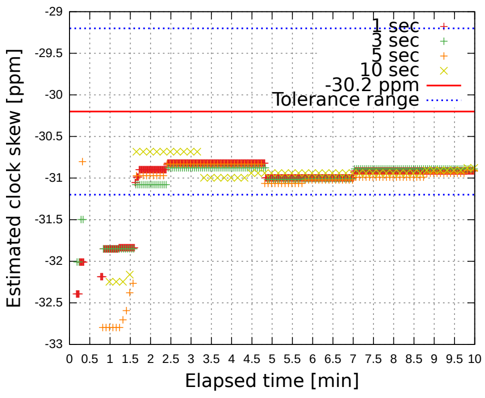

Figure 3: Detail view on the clock skew estimation from the four largest data files. The estimated clock skew of the computer is in the desired ± 1 ppm tolerance range from the final clock skew of -30.2 ppm after about two minutes. The influence of the duration of the measurement dominates the number of packets.

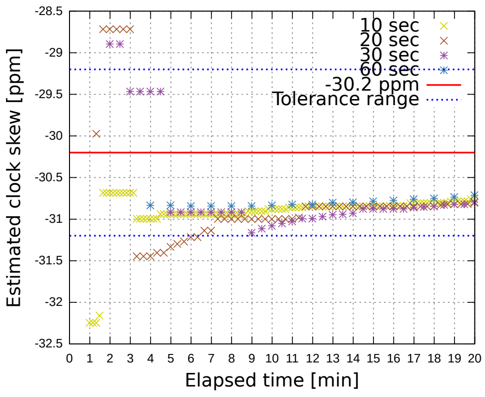

Figure 4: Detail view on the clock skew estimation from the four smallest data files. Even though the number of packets is smaller compared to Figure 3, the estimated clock skew of the computer is in the desired range after 4–6 minutes.

are in conformance with their observation of needing at least few minutes of data capture. However, we consider that the duration of the measurement is more important than the actual number of packets.

4 TIME STAMP PROPERTIES

This Section presents the properties that we found during the evaluation of the clock-skew-based identification. Firstly, there are inconsistencies between operating systems, especially inconsistency in TCP time stamps of Apple devices. Then, we argue that different sources of time stamps usually result into comparable clock skew estimates. Finally, we evaluate time changes and their influence on time stamps; any clock skew estimation can be jammed by the fingerprintee.

Table 1: Time stamp support in current OS (✓= enabled by default).

OS	TCP	ICMP	L7
Windows	Not enabled by default	Non-standard values	✓
Linux	✓	✓	✓
FreeBSD	✓	✓	✓
Mac OS X/iOS	Very high, frequently changing clock skew	Not supported	✓
Android	✓	✓	✓

4.1 Operating Systems

Although the clock-skew-based computer identification is applicable to all common operating systems (Microsoft Windows, Apple Mac OS X/iOS, Linux/Android), the default behaviour of the clock skew generating algorithms differs (Kohno et al., 2005; Sharma et al., 2012).

The Apple operating systems (Mac OS X and iOS) do not support ICMP time stamps anymore. In addition, computations from TCP time stamps result into very high clock skew estimations (hundreds or thousands of ppm). Moreover, the estimations are not stable and clock skew changes on unknown occasions. We closely investigated at least 5 different devices (including Mac Book Air, iPad, Mac Mini) and all of them behaved in a similar way (see Figure 5 for a result of one of the measurements). Note that the upper bound method of all offset points yields incorrect results. Instead, we detect a change in the clock skew and compute upper bounds segments, each for a period with constant clock skew.

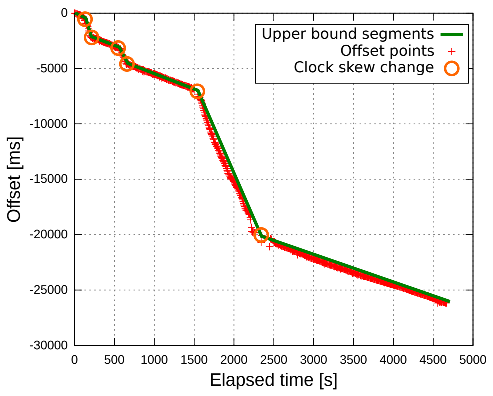

Figure 5: An example of unstable clock skew of devices running Mac OS X 10.8. Apple iOS behaves similarly.

None of the previous work reported similar behaviour, therefore, we tried to identify the source of the jamming. In order to exclude a bug in the software, we examined the packet traces manually. We indeed learnt that the computed clock skew is valid. In addition, we tested a virtual machine running Mac OS X that exhibited similar properties. In contrast, Windows running on a Mac mini generated time stamps with reasonable clock skew.

To investigate possible sources of the jamming, we tried experiments with controlled CPU load, installed new updates, used the computer or left it running without any user input. Nevertheless, we did not isolate the cause of the jammed time stamp values as the clock skew was changing in an unpredictable way in both Mac OS X and iOS.

The properties of the clock skew measured from other operating systems and other sources of time stamps were in expected range. The support of time stamps is summarized in the Table 1.

4.2 Time Stamp Sources

Our experiments with various time stamp sources show that estimations of clock skew computed from different time stamp sources generated by a single computer result into the same clock skew estimate. Figure 6 provides an example of a measurement of a Linux computer. Packet time stamps from ICMP, TCP as well as time stamps generated by JavaScript code were considered. All three sources converged to a clock skew estimate of -30.2 ppm.

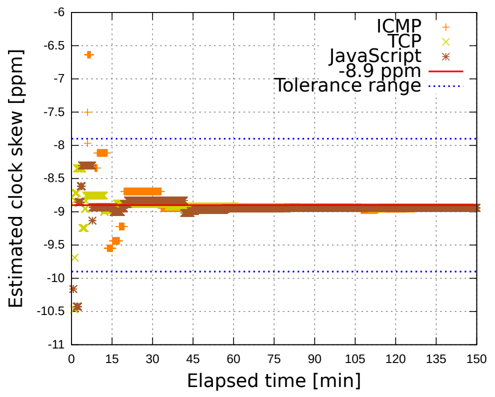

Figure 6: Clock skew estimated from different time stamp sources produced by the same computer converge to the same value.

However, as discussed above, computers running Apple operating systems are an exception. The clock skew computed from time stamps generated on the application level has the expected properties while it is impossible to compute a reasonable estimation from TCP time stamps (see Figure 7).

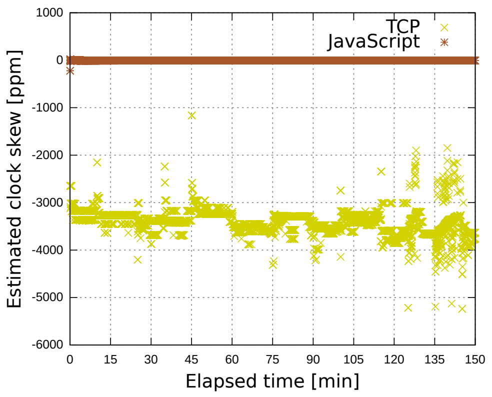

Figure 7: Clock skew estimated from different time stamp sources produced by Apple operating system. A reasonable clock skew estimation cannot be computed from TCP time stamps.

4.3 Time Synchronisation

Figure 8 depicts a measurement of a Linux fingerprintee running a utility called *ntpdate*. When *ntpdate* is started, it computes the offset between local time and the time advertised by NTP servers (Mills et al., 2010). If the time difference between the reference server and local time is lower than a threshold (0.5 s by default), *ntpdate* calls function *adjtime()*. Consequently, the system slowly compensates the time difference so that the internal clocks are synchronized to the value advertised by NTP without a big change in of time value. Then, the computer returns to the original value of system clock period and *ntpdate* exits.

Whenever *ntpdate* detects that the difference between local time and the reference NTP server is bigger than the threshold (0.5 s by default), it calls the function *settimeofday()*, which fixes the local time at once. TCP time stamp values are not affected in any way (see Figure 9) because TCP time stamps carry the time since the Linux kernel started, which is not influenced by the *settimeofday()*. ICMP and usually application-layer-generated time stamps reflects the local time. Hence, the call of the *settimeofday()* is visible for a third party observer, see Figure 9.

The experiments with other systems revealed that time stamps generated on application level are always affected by NTP as applications do not access the system clock directly, but they access the same time that is typically visible to the user.

As a result, similarly to application layer time stamps on Linux machines being affected by usage

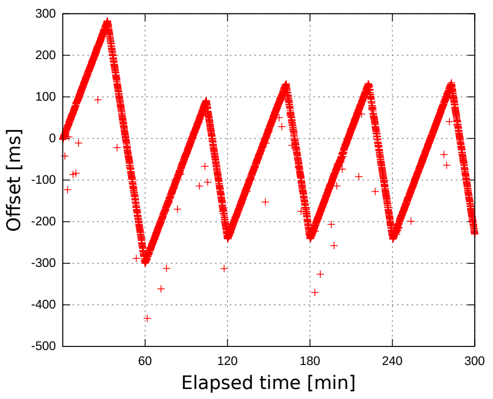

Figure 8: Time stamp values generated by a recent Linux kernel are influenced by time synchronisation of *adjtime()*. In this case, time was synchronised by a utility called *ntpdate* every hour. The graph shows different clock skew in the periods when ntpdate was trying to fix the clock (e.g. 146 ppm, the specific value varies) and periods without time synchronisation (about −354 ppm).

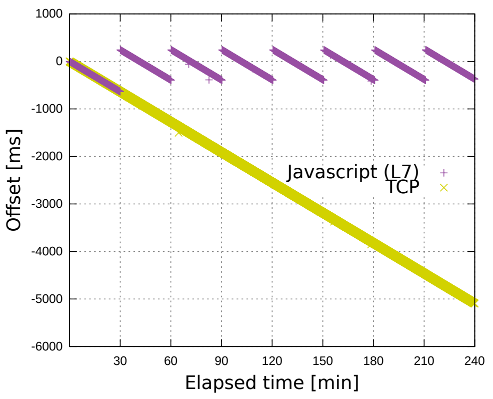

Figure 9: The clock skew of a computer with clock changed by *settimeofday()* every 30 minutes. The computer keeps its clock skew of about −354 ppm. While a TCP fingerprinter does not see any change in the clock skew, a fingerprinter of the time stamps sent by JavaScript code sees a shift in the time stamp value after every call of *settimeofday()*.

of *settimeofday()*, every time a user or a script modify the system time, the application layer time stamps are affected (see Figure 10). This can be exploited by a privacy-seeking user who can prevent the estimation of the clock skew by modifying his or her system clock more often than is the minimal observation period discussed in Section 3.

BSD 9.2 hosts are also affected by NTP. In contrast, TCP time stamps generated by Windows hosts are not influenced by NTP. Our experiments suggest that TCP time stamps produced by Apple devices are not affected by NTP. However, as discussed above, TCP time stamps of Apple devices are not stable enough to compute a reasonable clock skew.

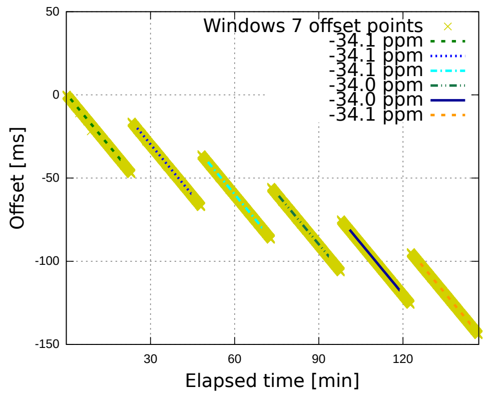

Figure 10: Clock skew measurement of a machine running Windows 7. Although the clock value was periodically modified, the clock skew can be estimated for periods of stable clock skew.

5 CLOCK SKEW DISTRIBUTION

For our final test, we focused on real network monitoring. We mirrored the traffic going through the network link that connects our faculty with the University network. From this traffic, we filtered TCP segments with time stamp options and estimated the clock skew for each recognized address. The monitored computers included:

- laboratory computers,
- desktops and laptops of the faculty staff,
- mobile devices connected to the faculty Wi-Fi,
- faculty servers,
- remote servers accessed by the above mentioned computers,
- remote clients that connected to local servers.

We focused only on TCP time stamps, we did not consider Windows clients because they do not send TCP time stamps by default. In addition, for privacy reasons, we did not look at the specific cases in details and we did not compare the gathered information to external sources. We observed 350–649 hosts during working hours and 120–170 hosts during night. The goal of this experiment was to gather more information about the feasibility of the clock-skew-based identification in real network environment.

The fingerprinter was not rebooted for months and during this period, it synchronised its clock through NTP. Hence, its clock tick period was stable during the experiment. Consequently, the measured values of the computers are likely very close to the correct clock skew compared to the universal time.

Firstly, we were interested in the clock skew distribution. With hundreds of devices, the clock skew needs to be spread over a large ppm space so that every single computer is identifiable. Since the distribution of clock skew did not change significantly during the experiment, let us focus on a snapshot of the network as an example — an afternoon of a working day when 646 hosts were detected in the network. The histogram of all estimated clock skews in the network at that time, displayed in Figure 11, shows that majority of addresses have clock skew close to zero.

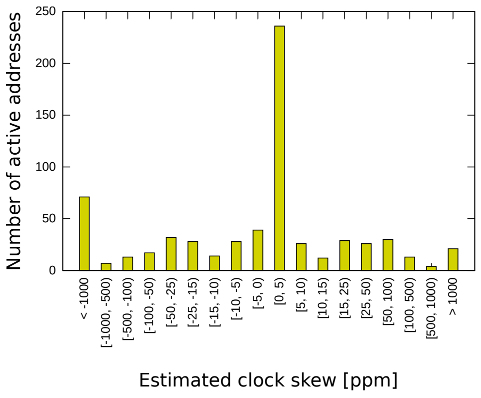

Figure 11: Histogram of clock skew distribution in real-network. Note that the closer a bin is to zero ppm the smaller clock skew range it covers.

The addresses with very high (over 1000 ppm) or low (less than -1000 ppm) clock skew showed similar properties to the Apple operating systems in our laboratory, e.g. the clock skew was not stable (see Section 4 for details about TCP timestamps of Apple devices). Therefore, we will not consider these addresses in the rest of this Section.

We observed two phenomena in the data. Firstly, considerable amount of estimated clock skew is very close to 0 ppm. The maximal amount of IP addresses in the 0 ± 1 ppm range was 223. It means that 34.5 % of computers were not distinguishable from the fingerprinter. We believe that this is caused by the time synchronisation.

The second phenomenon is closely related to the observation made by (Lanze et al., 2012). The Figure 12 reports the number of indistinguishable computers for each estimated clock skew (x-axis); the hosts employing time synchronisation are not plotted. Most of the clock skew values are distributed in a relatively close range around 0 ppm even when the estimation is not spoiled by NTP.

To evaluate if the observed clock skew distribution is only related to our network with, for instance, a large amount of similar computers in the laboratories, we consulted also other related literature. We were able to get in touch with authors of (Sharma et al., 2012) and they did not observe this phenomenon. We

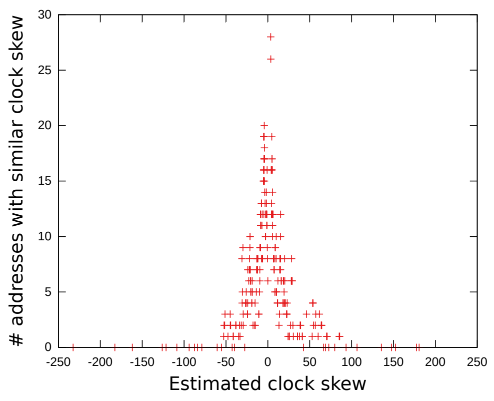

Figure 12: The estimated clock skew and number of devices with similar clock skew in the range of -250 ppm to 250 ppm, the effect of NTP is omitted in the figure.

did not receive any reply from the authors of (Ding-Jie Huang et al., 2012), however, the positioning of the estimated clock skew in Figure 10 of (Ding-Jie Huang et al., 2012) indicates that clock skew of devices 15–85 were closer to each other in comparison to the other devices. Indeed, our data plotted in this fashion would look alike.

6 DISCUSSION

There is a considerable amount of external factors that hinders the unique identification for a third party.

Firstly, an external entity does not have precise data from hardware clocks. When a clock skew is estimated from network data, time stamps are affected by the following:

- The operating system: 1) the clock skew differs when a computer is rebooted to different operating system and 2) the operating system impacts the availability and quality of the generated time stamps, e.g. Windows clients do not send TCP timestamps by default or the clock skew cannot be estimated from TCP time stamps of Apple operating systems.
- Time shifts: the observed computer may synchronise its clock with precise time sources from Internet, or, the user may directly change the clock value and jam the external clock skew estimation.
- Delays in the network: The more jitter in the network between the fingerprintee to the fingerprinter the more packets are needed to estimate the correct clock skew.

On top of the aforementioned external factors, all other factors that are present locally affect clock skew, e.g. temperature or voltage.

Secondly, the goal of clock manufacturers is to make as precise clock as possible. Hence, the expected clock skews are bound to be close to 0 ppm. The observation of real network suggests that the clock skew of the majority of devices is in the range from -50 ppm to 100 ppm.

Finally, the end user is in a good position to jam the in-built clock skew effect on time stamps sent from his or her computer. TCP time stamps are not present in TCP segments in connections originating from Windows machines. Hence, for privacy reasons, disabling TCP time stamps is a valid choice. The user space sources of time stamps are affected by time synchronisation or manual changes of clock values. A privacy seeking user may consider permanent synchronisation with precise time servers or frequent changes of the clock value of his or her computer. The former hides the user among all other synchronised machines, the latter prevents a fingerprinter to estimate the clock skew.

Whereas in a smaller network the identification of all devices is possible provided that the users does not take any counter-measures, in bigger network the probability of two or more computers having similar clock skew increases. Therefore, any fingerprinting in bigger networks needs to take into account false positives.

Other authors published similar methods to detect computer identity, e.g. browser fingerprinting (Eckersley, 2010) based on HTTP headers, installed fonts, etc. Browser fingerprinting works only in combination with HTTP and also suffers from false positives, e.g. all computers in our laboratory are identified to be the same since they have been cloned from the same image. In contrast, their clock skew differs. The combination of the two methods can limit the number of false positives. However, we did not investigate this in detail.

Models of user behaviour (Banse et al., 2012; Herrmann et al., 2012; Kumpošt, 2008) rely on specific communication patterns of monitored users. The disadvantage of these models is the need to track users for a long time, e.g. one day (Banse et al., 2012). Often, the models have difficulties when users regularly change their IP addresses (Herrmann et al., 2012). As the precision depends on user behaviour, they cannot identify users without stable communication patterns.

The method can be used as a weak proof of a computer not being present in a specific network. When a user is prevented to influence the clock skew, e.g. when he or she does not have root access to the system, whenever a previously known clock skew is not observed on any IP address, the computer does not communicate via the network.

7 CONCLUSION

Since (Kohno et al., 2005) established the field of clock skew estimation from network traces, it expanded into various areas, such as identification of anonymous services (Zander and Murdoch, 2008), wireless networks (Jana and Kasera, 2010) and web user identification (Ding-Jie Huang et al., 2012). The basic idea behind the clock-skew-based identification seems to be valid especially in small networks but some authors already warned that the distribution of clock skew is not ideal for unique identification (Ding-Jie Huang et al., 2012).

This study revisited the findings from previous research of the clock-skew-based identification. This paper argues that the soundness of the clock skew estimation does not depend merely on the amount of packets as previous studies suggested. Instead, the duration of the measurement is more important as the effects of the built-in clock skew increases with time.

During the study of the impact of operating systems on clock skew, we discovered an irregularity of time stamps originating from Mac OS X and iOS. The irregularity was clearly visible both during laboratory experiments, with devices under our control, and during the real network experiment, with devices of other users.

The study of time synchronisation and manipulation unveiled that clock skew can be influenced by NTP which may prevent to estimate correct clock skew.

The real network experiment revealed considerable difficulties of clock-skew-based identification in large networks. In our network, the detected clock skew of almost all devices is in the range between -50 ppm and 100 ppm with majority much closer to 0 ppm. Time synchronization was employed by up to 40 % of observed devices as their clock skew was very close to 0 ppm and they were not distinguishable between each other.

Passive TCP-level fingerprinting of Windows machines is not possible, Apple operating systems have unstable clock skew and clock skew of Linux and BSD machines is affected by running NTP. Application level time stamps are always affected by time modifications.

ACKNOWLEDGEMENTS

This work is a part of the project VG20102015022 supported by Ministry of the Interior of the Czech Republic. It was also supported by the project FIT-S-14-2299 of Brno University of Technology.

REFERENCES

Banse, C., Herrmann, D., and Federrath, H. (2012). Tracking users on the internet with behavioral patterns: Evaluation of its practical feasibility. In *Information Security and Privacy Research*, volume 376, pages 235–248. Springer Berlin Heidelberg, DE.

Ding-Jie Huang, Kai-Ting Yang, Chien-Chun Ni, Wei-Chung Teng, Tien-Ruey Hsiang, and Yuh-Jye Lee (2012). Clock skew based client device identification in cloud environments. In *Advanced Information Networking and Applications*, pages 526–533.

Eckersley, P. (2010). How unique is your web browser? In *Privacy Enhancing Technologies*, volume 6205 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin Heidelberg, DE.

Herrmann, D., Gerber, C., Banse, C., and Federrath, H. (2012). Analyzing characteristic host access patterns for re-identification of web user sessions. In *Information Security Technology for Applications*, volume 7127, pages 136–154. Springer Berlin Heidelberg, DE.

Jacobson, V., Braden, B., and Borman, D. (1992). *RFC 1323 TCP Extensions for High Performance*.

Jana, S. and Kasera, S. (2010). On fast and accurate detection of unauthorized wireless access points using clock skews. *IEEE Transactions on Mobile Computing*, 9(3):449–462.

Kohno, T., Broido, A., and Claffy, K. (2005). Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2):93–108.

Kumpošt, M. (2008). *Context Information and User Profiling*. PhD thesis, Masaryk University, CZ.

Lanze, F., Panchenko, A., Braatz, B., and Zinnen, A. (2012). Clock skew based remote device fingerprinting demystified. In *Global Communications Conference*, pages 813–819.

Mills, D. L., Martin, J., Burbank, J., and Kasch, W. (2010). *RFC 5905 Network Time Protocol Version 4: Protocol and Algorithms Specification*.

Murdoch, S. J. (2006). Hot or not: Revealing hidden services by their clock skew. In *Computer and Communications Security*, pages 27–36, New York, NY, USA. ACM.

Polčák, L., Jirásek, J., and Matoušek, P. (2013). Comments on ”Remote physical device fingerprinting”. *IEEE Transactions on Dependable and Secure Computing*. Pre-prints.

Sharma, S., Hussain, A., and Saran, H. (2012). Experience with heterogenous clock-skew based device fingerprinting. In *Workshop on Learning from Authoritative Security Experiment Results*, pages 9–18. ACM.

Zander, S. and Murdoch, S. J. (2008). An improved clock-skew measurement technique for revealing hidden services. In *Proceedings of the 17th Conference on Security Symposium*, pages 211–225, Berkeley, CA, USA. USENIX Association.