Journal of Universal Computer Science, vol. 21, no. 9 (2015), 1210-1233
submitted: 23/9/14, accepted: 18/8/15, appeared: 1/9/15 © J.UCS

Clock-Skew-Based Computer Identification: Traps and Pitfalls

Libor Polčák
(Faculty of Information Technology, Brno University of Technology,
Božetěchova 2, 612 66 Brno, Czech Republic
ipolcak@fit.vutbr.cz)

Barbora Franková
(Faculty of Information Technology, Brno University of Technology,
Božetěchova 2, 612 66 Brno, Czech Republic
xfrank08@stud.fit.vutbr.cz)

Abstract: Each clock has built-in deficiencies since the manufacturing process is not precise on atomic level. These inaccuracies cause each clock to drift in a unique way. Clock skew has been already studied and used to identify computers. Based on the previous research in clock-skew-based identification, this paper provides a summary of use cases and methods for clock-skew-based identification. Nevertheless, the main contribution of the paper is following: (1) A formal evaluation of the requirements for precise clock skew estimations. The formal approach is accompanied with an empirical study of 24,071 clock skew measurements. (2) A method that links IPv4 and IPv6 addresses of a single computer. (3) A scenario, during which a malicious attacker mimics clock skew of another computer and consequently, for example, penetrates through authentication mechanisms considered during previous research. (4) Even though the real network observations expose that current precision in clock skew estimation is not sufficient to uniquely identify devices in moderately-sized network, some IPv4 and IPv6 addresses can be linked based on unique clock skew shifts of a computer, for example caused by a running NTP daemon.

Key Words: Device fingerprinting, clock skew, security, counter-measures, IPv6.
Category: C.2.3

1 Introduction

A computer clock is typically controlled by a crystal oscillating with a specific frequency. However, due to physical limitations of the manufacturing process, each crystal is unique on the atomic level. The differences result in a small deviation of the frequency of the crystal oscillation, and consequently, cause a small inaccuracy in the time measurement. The inaccuracy is called clock skew.

During remote clock-skew-based computer identification, an observer (a *fingerprinter*) gathers time stamps of computers to be identified (*fingerprintees*) with the goal of unique identification of all or specific computers. Remote clock-skew-based computer identification was first presented by [Kohno et al., 2005] with applications in location tracking, device detection behind a network address translator, and honeynet detection. Later, other applications followed, e.g. deanonymization [Murdoch, 2006], rogue access point detection [Jana and Kasera, 2010], and multi-factor authentication [Huang et al., 2012].

This paper pursues the ideas that originally appeared in our previous paper [Polčák and Franková, 2014]. The previous paper argued about the minimal requirements to compute sound clock skew estimates and unveiled that unique identification in real network is hardly possible. In this paper, we provide the formal evaluation of requirements for sound clock skew estimates. The evaluation supports the claims raised in the previous paper [Polčák and Franková, 2014]. The precision of the estimates does not solely depend on the number of packets as previous studies suggested [Sharma et al., 2012]. Instead, the duration of the measurement is more important as it allows a fingerprinter to detect time stamps that compensate the variable time stamp delivery latency. As a side effect, longer fingerprinting is correlated with higher number of packets. Higher number of packets can reveal more information about processing and network delay. To demonstrate the requirements, this paper provides a real-network study consisting of 24,071 clock skew estimation samples.

Another contribution of this paper is in the exposure of a method to mimic clock skew of another computer. Applying this method, an attacker may confuse the multi-factor authentication discussed in the past [Huang et al., 2012]. Another applications of the method allows to protect a set of computers from reconnaissance attacks based on clock skew estimation [Kohno et al., 2005] or a single laptop can mimic a different clock skew each time the laptop reconnects to the network and consequently limit the previously published location tracking algorithm [Kohno et al., 2005].

The final contribution of this paper is the applicability of clock-skew-based identification for linking IPv4 and IPv6 addresses of a single computer. Our results confirm the expectation that the clock skew is not influenced by the IP family. Therefore, it is possible to link the IPv4 address and all IPv6 addresses of one computer. The laboratory results showed that the method is viable. However, a real network deployment can fail since the clock skew is not unique enough to identify several hundreds of computers [Polčák and Franková, 2014]. Nevertheless, the changes in clock skew are observable by a long-term fingerprinter if he or she has enough time stamp samples spread in time. Consequently, the long-term fingerprinter can link IPv4 and IPv6 addresses based on the synchronous changes of clock skew measured for all addresses of the same computer.

This paper is organized as follows. Section 2 introduces the clock-skew-based identification and overviews the previous research. Section 3 classifies fingerprinters, lists their use cases for clock-skew-based identification, and overviews time stamp sources. The conditions for computation of sound estimates are derived in Section 4. Section 5 reveals a possibility to spoof clock skew by an attacker trying to mimic a computer of a victim. Section 6 demonstrates how to use clock skew to reveal all IPv6 addresses of one computer. Section 7 covers real network deployment and the study of clock skew distribution in real network. Section 8 discusses the impact of this paper on clock-skew based identification and Section 9 concludes the paper.

2 Clock skew estimation

Kohno et al. introduced the original idea behind clock skew computation to identify network devices [Kohno et al., 2005] as follows. 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 the difference between two clocks: $\mathrm{off}_{C,D}(t) \equiv \mathrm{R}_C(t) - \mathrm{R}_D(t)$. 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, often referred as *parts per million* (ppm).

Consider C to be the clock of the fingerprinter and D to be the clock of the fingerprintee as depicted in Figure 1. R_D is not observable by the fingerprinter, instead, it sees packets marked with time stamps delayed by $\epsilon(t)$, i.e. the delay observed at time t. The delay $\epsilon(t)$ is composed of the processing time at both the fingerprinter and the fingerprintee and the network delay. If ϵ were constant, the first derivative of $\mathrm{off}^{\epsilon}{}_{C,D} \equiv \mathrm{R}_C(t) - \mathrm{R}_D(t - \epsilon(t))$ would have been equal to the first derivative of $\mathrm{off}_{C,D}$. Unfortunately, ϵ is not a constant.

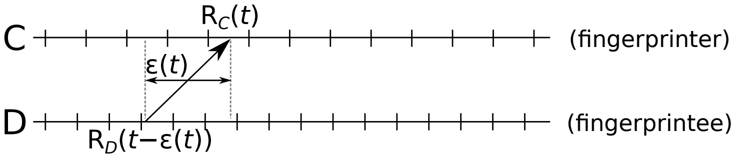


Figure 1: Each time stamp of the fingerprintee is delayed by the network and the network stacks of the end hosts.

Let us represent observed timestamps 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, i.e. $\mathrm{R}_C(t) - \mathrm{R}_C(t_{\mathrm{start}})$; and y is the observed offset $\mathrm{off}^{\epsilon}{}_{C,D}(t)$. 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 2 for an example.

The original paper [Kohno et al., 2005] studied the advantages and disadvantages of the clock-skew-based identification. Kohno et al. used two sources of

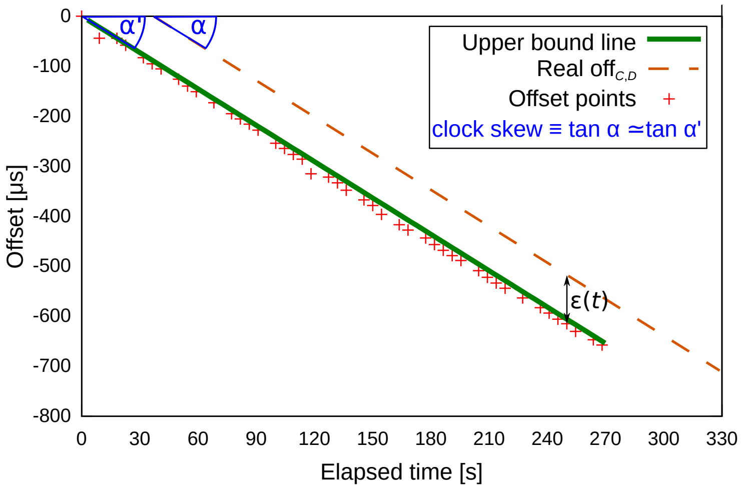


Figure 2: Clock skew estimation.

time stamps: time stamps from TCP can be collected passively whereas gathering time stamps from ICMP needs active probing.

Later, time stamps carried on the application layer were considered [Murdoch, 2006,Zander and Murdoch, 2008,Huang et al., 2012]. Zander and Murdoch computed clock skew from time stamps present in the HTTP protocol. HTTP defines Date header that typically contains the time stamp reflecting the time when the reply was generated. Usually the resolution is 1 second and therefore the frequency is 1 Hz. While HTTP 1.0 recommended Date headers, since HTTP 1.1, the Date header is a mandatory header in almost all replies (except some 5xx errors and 1xx replies). The dominant version of HTTP is 1.1 so the method is generally applicable. However, the small frequency might result in a quantisation error [Zander and Murdoch, 2008] of up to 1 second.

Nevertheless, the quantisation error can be compensated [Zander and Murdoch, 2008] with synchronized sampling. A fingerprinter employing synchronized sampling tries to synchronize HTTP request probes with clock ticks of the fingerprintee. As a result, active fingerprinters can remove the quantisation error to determine clock skew even from low frequency time stamp sources.

Besides HTTP, many other application layer protocols carry time stamps, e.g. XMPP, SMTP, and RTP. Usually, the resolution of these time stamps is in seconds, therefore, the measurement is spoiled by quantisation error if the fingerprintee does not remove it, e.g. by synchronized sampling [Zander and Murdoch, 2008].

Ding-Jie Huang et al. [Huang et al., 2012] employed 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 login web page contains custom JavaScript code that periodically sends time stamps from the fingerprintee to the fingerprinter.

Sharma et al. improved the detection by introducing a batch of ICMP packets to compensate for the latency and losses on the network path [Sharma et al., 2012]. We did not consider using this approach for the following reasons: (1) our focus was also on passive detection, (2) ICMP time stamps are not supported by Windows in a standard way, (3) ICMP time stamps are disabled by default in Apple operating systems, and (4) the successor of ICMP for IPv6, ICMPv6, does not define ICMPv6 Time stamp Request and Reply.

Although [Huang et al., 2012] improved the clock skew estimation by linear regression and other techniques, 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. However, in this paper, we derive a formula that quantifies the error of the estimation. This formula can be used to reduce the ambiguity of clock skew estimates during long-term measurements.

One of our previous works [Polčák et al., 2014] reported that time shifts of Linux hosts are propagated to TCP time stamps, e.g. by NTP-enabled hosts. Our following work [Polčák and Franková, 2014] studied the NTP influence on other sources of time stamps. Time stamp inserted in user space are influenced by NTP. The paper also presented that in the meantime, NTP-enabled BSD hosts also propagate clock changes to TCP time stamps.

The previous paper [Polčák and Franková, 2014] already argued about the minimal requirements to compute clock skew estimates. In this paper, we extend the discussion by formal examination of the important factors influencing the quality of a clock skew estimate. Additionally, this paper presents a study of more than 24,071 samples of clock skew estimation to validate the examination.

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.

3 Use cases for clock-skew-based identification

The previous section shows that clock-skew-based identification has already been tested for different applications. Specific applications have specific requirements on the clock-skew-based identification. This section provides a classification of fingerprinters that can employ clock skew based identification, the applications and the methods to obtain time stamps.

A fingerprinter can be either passive or active.

- A passive fingerprinter monitors network traffic and detects time stamps carried in packets, mostly in protocol headers, e.g. TCP [Kohno et al., 2005] or HTTP. As a passive fingerprinter does not modify network traffic, his or hers activities are undetectable by fingerprintees.

- In contrast, an active fingerprinter injects time stamp probe packets to the network that yields him or her additional time stamps, e.g. ICMP [Kohno et al., 2005], JavaScript generated time stamps [Huang et al., 2012], or additional information during synchronized sampling [Zander and Murdoch, 2008]. The main drawback of the active fingerprinting is the presence of the additional traffic that consumes network bandwidth and creates a risk of a fingerprintee revealing the additional traffic.

Depending on the time available to provide clock skew estimates, two possible scenarios arise:

- A quick (rapid) clock-skew estimation allows an attacker to quickly compute a clock skew estimation [Huang et al., 2012]. The goal is to identify the device originating the traffic as fast as possible.
- During long-term measurements, a fingerprinter does not need to have the results as fast as possible. Consequently, the fingerprinter can observe long term clock skew development, such as small shifts in clock skew caused by temperature changes [Murdoch, 2006].

Based on the previous two lists, there are four types of possible fingerprinters, each type is suitable for different type of applications of clock-skew-based identification.

1. A passive rapid fingerprinter can quickly identify a device during targeted surveillance, e.g. for lawful interception (during which the observer cannot alter the observed traffic). Another application is the identification of rogue access points [Lanze et al., 2012], during which a user trying to access a wireless network confirms the identity of the access point.
2. A passive long-term fingerprinter can monitor changes in clock skew caused by NTP or temperature changes. Consequently, the fingerprinter can learn all IP addresses used by a specific computer (see Section 6), track location of a computer [Kohno et al., 2005], learn geographical location [Murdoch, 2006] or reveal computers behind network address translator [Kohno et al., 2005].
3. An active rapid fingerprinter can use clock-skew-based identification during multi-factor authentication [Huang et al., 2012] or to improve the estimates [Sharma et al., 2012] compared with rapid passive fingerprinter.

4. An active long-term fingerprinter can improve the quality of the clock skew estimates compared to both active rapid fingerprinter and passive long-term fingerprinter. Applications include deanonymization [Zander and Murdoch, 2008], virtual PCs and honeypot detection [Kohno et al., 2005].

So far, several sources of time stamps have been identified:

- TCP time stamps are one of the TCP options [Jacobson et al., 1992, Borman et al., 2014]. TCP time stamps are present in each TCP segment header when a client and a server negotiate this option during the initial TCP phase. Since the original paper [Kohno et al., 2005], it is known that TCP time stamps are not generated by Windows devices by default. We evaluated that this is valid to this date and Windows 8.1 (and also Windows 10, pre-release build 10074) still does support neither RFC 7323 nor the original RFC 1323 by default. Kohno et al. injected time stamp into the first SYN packet, and consequently forced Windows clients to include TCP time stamps into consecutive segments. Therefore, an active fingerprinter can trick Windows client to add TCP time stamps. Linux, Mac OS X, iOS, and FreeBSD include TCP time stamps by default. However, Linux and FreeBSD TCP time stamps are influenced by NTP [Polčák et al., 2014, Polčák and Franková, 2014] and Mac OS X and iOS timestamps are influenced by unknown error [Polčák and Franková, 2014] (valid also in Mac OS X 10.7.5 and iOS 8.3).
- ICMP defines Timestamp Request that asks a remote host to respond with its current time stamp. Hence, ICMP time stamps are available only for an active fingerprinter. The downside of the ICMP-based measuring is the non-uniform implementation in current operating systems [Polčák and Franková, 2014]. Furthermore, ICMP Timestamp Request is defined in IPv4 but it is not available in IPv6.
- HTTP, XMPP, and several other application layer protocols add time stamps to all or specific messages. A passive fingerprinter can observe these time stamps. However, these time stamps have usually the frequency of 1 Hz and suffer from high quantisation error [Murdoch, 2006]. An active fingerprinter can synchronize probe requests with clock of the fingerprintee and effectively remove the quantisation error [Zander and Murdoch, 2008].
- An adversary capable of executing code on the fingerprintee (e.g. via AJAX – JavaScript code embedded to a web page [Huang et al., 2012]) can enforce the fingerprintee to send time stamps either directly to the fingerprinter or via a link that the fingerprinter observes.
- IEEE 802.11 (Wi-Fi) networks need synchronised clocks between an access point and all hosts to maintain basic functionality such as frequency hop-

ping. Access points send periodical beacons that a passive fingerprinter can monitor [Jana and Kasera, 2010, Lanze et al., 2012].

Table 1 summarizes the time stamp sources, their type, and the frequency of the clock value increments (clock ticks).

Method	Type	Frequency
ICMP	Active	1 kHz
TCP	Passive	10-1000 Hz (OS-dependant)
Application layer protocols	Active or passive	Method/OS-dependant
Enforced time stamps	Active	Method/OS-dependant
IEEE 802.11 beacon frames	Passive	1 MHz

Table 1: Comparison of time stamp sources.

4 Quick availability of estimates

Before employing clock-skew-based computer identification, it is necessary to be aware of the stability of clock skew and the accuracy of the computed estimates. This section revisits the discussion [Kohno et al., 2005, Sharma et al., 2012, Polčák and Franková, 2014] about the requirements for accurate estimation of the clock skew of a specific computer for rapid clock-skew-based identification.

This section proves that the quality of clock skew estimates depends on the duration of the measurement and the stability of the time stamp observation delay. Nevertheless, the number of observed packets influence the quality of the estimate in two ways: 1) the longer the fingerprinting lasts the more packets are available and 2) the more packets are available the more probable it is to get time stamps shifted with such similar delay ϵ, so that the time difference between the two packets forming the upper bound compensates for the difference in the delay ϵ.

4.1 Formal evaluation

For simplicity, let us denote $\text{off}_{C,D} \equiv \text{off}$ and similarly ${\text{off}^{\epsilon}}_{C,D} \equiv \text{off}^{\epsilon}$ in the following text as both functions refer to the offset between the clocks of the fingerprinter and the fingerprintee.

By the definition of the upper bound, there are at least two offset points located on the upper bound. Let us refer one of these points as an offset point X. Its coordinates are $X = [t_X, \text{off}^{\epsilon}(t_X)]$ where t_X is the observation time.

In addition, let us denote the amount of time elapsed during period T (of the national standards *true time*) on clock C as $\mathrm{E}_C(T)$. According to the equation 1, the y-coordinate of X equals to $\mathrm{off}(t_X) - \mathrm{E}_D(\epsilon(t_X))$.

$$\begin{aligned} \mathrm{off}^{\epsilon}(t) &= \mathrm{R}_C(t) - \mathrm{R}_D(t - \epsilon(t)) = \\ &= \mathrm{R}_C(t) - \mathrm{R}_D(t) - \mathrm{E}_D(\epsilon(t)) = \\ &= \mathrm{off}(t) - \mathrm{E}_D(\epsilon(t)) \end{aligned} \tag{1}$$

Figure 3 depicts the geometry of a clock skew estimation. Consider the two offset points A', B' located on the upper bound $\mathrm{b}(t)$. The offset points were observed at time t_1 and t_2, hence $\mathrm{b}(t_1) = \mathrm{off}(t_1) - \mathrm{E}_D(\epsilon(t_1))$. The points A', B', and the point $C' = [t_2, \mathrm{b}(t_1)]$ form a right triangle. In addition, consider the counter image triangle defined by points A, B, and C, such that A and B is located on the line representing real offset, $\mathrm{off}(t)$, of the clocks. Hence, the points A and B represent the real offset between the clock of the fingerprinter and the fingerprintee at the time t_1 and t_2 that is not biased by the delay ϵ.

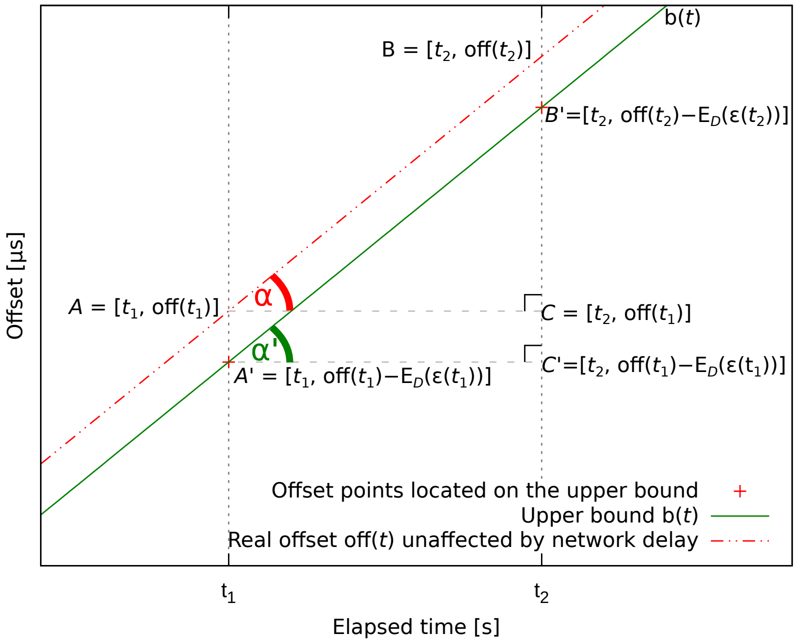

b(t)
B = [t2, off(t2)]
B'=[t2, off(t2)−ED(ε(t2))]
Offset [µs]
A = [t1, off(t1)]
α
α'
C = [t2, off(t1)]
A' = [t1, off(t1)−ED(ε(t1))]
C'=[t2, off(t1)−ED(ε(t1))]
Offset points located on the upper bound +
Upper bound b(t)
Real offset off(t) unaffected by network delay
t1
t2
Elapsed time [s]

Figure 3: Outline of clock skew computation used for error estimation.

The formula 2 defines the observed clock skew whereas the real clock skew is defined by the formula 3.

$$\mathrm{s}_{\mathrm{observed}} \equiv \tan\alpha' = \frac{\mathrm{off}(t_2) - \mathrm{off}(t_1) + \mathrm{E}_D(\epsilon(t_1)) - \mathrm{E}_D(\epsilon(t_2))}{t_2 - t_1}, \tag{2}$$

$$\mathrm{s} \equiv \tan\alpha = \frac{\mathrm{off}(t_2) - \mathrm{off}(t_1)}{t_2 - t_1}. \tag{3}$$

Combining equations 2 and 3, one can derive equation 4 that expresses the dependency of the observed clock skew on the real clock skew and the error introduced by the observed volatile latency ϵ.

$$\mathrm{s}_{\mathrm{observed}} = s + \frac{\mathrm{E}_D(\epsilon(t_1)) - \mathrm{E}_D(\epsilon(t_2))}{t_2 - t_1}. \tag{4}$$

Let us denote $\Delta t \equiv t_2 - t_1$ and $\Delta\epsilon \equiv |\,\mathrm{E}_D(\epsilon(t_1)) - \mathrm{E}_D(\epsilon(t_2))|$. Additionally, let us denote the expected precision of a clock skew measurement as P. To satisfy the precision requirement, the observed error (introduced by the formula 4) has to be lower than P as displayed in the formula 5.

$$\frac{\Delta\epsilon}{\Delta t} \le P. \tag{5}$$

In contrast with the expectation of needing 70 packets to estimate reliable clock skew [Sharma et al., 2012], the formula 5 shows that the quality of the estimation depends on the instability of the network latency $\Delta\epsilon$ and the elapsed time Δt. Note that Δt is weakly lower than the total duration of the measurement as Δt cannot be bigger than the measurement duration. Hence, a longer measurements can yield more precise estimates as a longer measurement can detect offset points with similar $\Delta\epsilon$ and bigger Δt.

Supposing that the fingerprinter observes only packets going through one network path, the latency introduced by wires and fibres is constant. Papagiannaki et al. studied the single hop delay [Papagiannaki et al., 2002]. Their conclusion is that *there is at least one packet that experiences no queueing in each one minute interval* on a hop. Additionally, each middle box delays a packet with minimal latency with certain probability. However, as their study is already more than 10 years old, their models for expressing the probability could have been invalidated by better routers and switches that does not experience the long delays that Papagiannaki et al. observed.

Therefore, we do not continue in quantifying the error. Based on the study of Papagiannaki et al., a higher number of packets can yield an upper bound for which the $\Delta\epsilon$ is low. Nevertheless, this effect is only indirect and additional effects have to be considered. For example, the latency ϵ depends on the network topology as each middle box adds to the total delay. In addition, network load or packet size [Papagiannaki et al., 2002] also influence the queueing time. Longer duration of a measurement Δt relaxes the conditions on the fluctuation of ϵ as shown in the formula 5 as well as additional packets that can experience lower ϵ.

Additionally, the software generating time stamps may increase the variance of ϵ when an already obtained time stamp value is delayed by a data processing algorithm that constructs the network message. For example, a user space code can construct each message based on a different number of kernel calls or there might be a garbage collector running that delays the execution of the program code.

Even more, computer clocks are updated by a constant value. The exact value depends on the underlying frequency of the clock source. This creates quantisation error and it is an important factor in the clock skew computation. Without synchronized sampling, it takes longer to compute the clock skew of a source with a frequency of 1 Hz than 100 Hz simply because the time that the clock value does not change contributes to the ϵ. Consequently, the increased instability of $\Delta\epsilon$ has to be compensated with bigger Δt to achieve the same precision. Nevertheless, an active fingerprinter can synchronize [Zander and Murdoch, 2008] probe packets with clock ticks at the fingerprinter to remove the quantisation error from ϵ.

4.2 Empirical evaluation

To examine the relation between fingerprinting duration and the number of packets to compute reliable clock skew estimates, we set up two experimental scenarios during which we monitored time stamps generated by our laboratory computers, running Red Hat Enterprise Linux 6.6 with Linux kernel 2.6.32.

In the first experimental scenario, we fingerprinted 18 computers. We run a Python script repeatedly calling the *time.time()* function. The script then sent the time stamp to the fingerprintee in a TCP flow with TCP time stamping enabled. TCP time stamps had a frequency of 1 kHz whereas the Python inserted time stamps had a frequency of 10 MHz. We included the very high Python-inserted time stamps to get as precise time stamps as possible. Since two successive calls of *time.time()* yields different value, we removed the quantisation error.

All computers generated time stamps with a Poisson distribution with expected number of requests between 1 and 10 per second. Time stamps were sent to the fingerprinter in HTTP requests, each yielding one Python-inserted time stamp and several TCP time stamps. The scenario simulated possibilities of a passive fingerprinter observing some kind of interactive web application with frequent communication with the web server. At the same time, it simulated the possibilities of an active fingerprinter generating time stamps in a user space application, e.g. for authentication purposes [Huang et al., 2012].

The second experimental scenario aimed on JavaScript-inserted time stamps in a Firefox browser. The fingerprinter repeatedly monitored 4 computers, each one was periodically (with a period between 0.1 s and 1 s) sending its current time stamp obtained from the *Date.getTime()* function. The function provides time stamps with 1 kHz frequency. To minimize the quantisation error, we let the JavaScript code to repeatedly obtain a new value from *Date.getTime()* while the returned value did not change. Therefore the code detected a clock tick and sent the time stamp to the fingerprinter. The second scenario studied the possibilities of an active fingerprinter capable of running JavaScript code in a user browser [Huang et al., 2012].

In total, we run 9,959 TCP experiments, 10,016 Python-inserted time stamp experiments, and 4,096 JavaScript-inserted time stamp experiments. During each experiment, we computed current clock skew of the computer based on the actual conditions (e.g. temperature). Then, we determined the number of sent packets and the elapsed time, after which the clock skew estimation did not leave the *final value* ±1 *ppm* interval. Note that in both scenarios, TCP time stamps contained quantisation noise as neither the fingerprinter nor the fingerprintees tried to detect ticks from TCP time stamps.

Figure 4 shows that hundreds of packets carrying TCP or Python-generated time stamps were needed in a majority of the experiments, some experiments required thousands of time stamps before the clock skew estimate converged to the ±1 ppm range of the expected value. We did not observe any special boundary associated with 70 time stamps as reported in previous research [Sharma et al., 2012]. Whereas 41.67 % of experiments with JavaScript-inserted already converged before detecting 70 time stamps, only 0.35 % of TCP experiments and only 0.28 % of experiments with Python-inserted time stamps required 70 time stamps or less.

Given a specific number of packets, the share of already converged experiments with Python-inserted and JavaScript-inserted time stamps were higher compared to the measurements based on TCP time stamps. Each HTTP request yielded several TCP time stamps but only one Python-inserted or JavaScript-inserted time stamp. Hence, there were a higher number of TCP time stamps in a fixed interval during each measurement. Therefore, a measurement consisting of a specific number of Python-inserted or JavaScript-inserted time stamps generally took longer than a measurement computing clock skew from the same number of TCP time stamps.

Figure 5 depicts the share of experiments converged to the *final value* ±1 *ppm* interval at a specific time after the start of the measurement. As obvious, Python-inserted time stamps took generally much longer to converge. 41.2 % of TCP experiments converged in 20 seconds, 70.6 % converged in 30 seconds and 97.3 % converged in one minute. Only 2.8 % of Python-inserted time stamp experiments

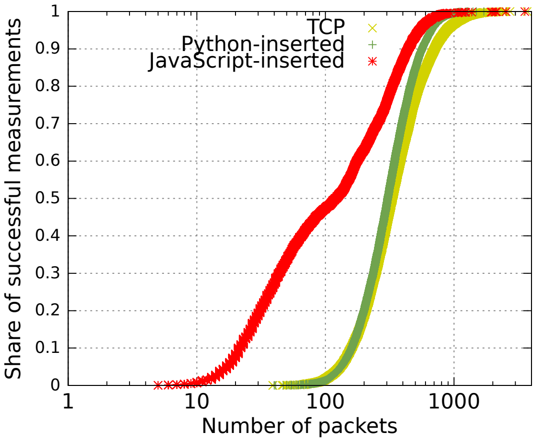


Figure 4: Experiments with Python-inserted and JavaScript-inserted time stamps required lower number of time stamps due to a higher frequency or lower quantisation error.

converged in one minute and 95.1 % converged in 6 minutes. JavaScript-inserted time stamp experiments have different development: 38.1 % of JavaScript-inserted time stamp experiments converged in 20 seconds, 53.4 % in one minute, 96.9 % in 3.5 minutes and 99.7 % converged in 6 minutes.

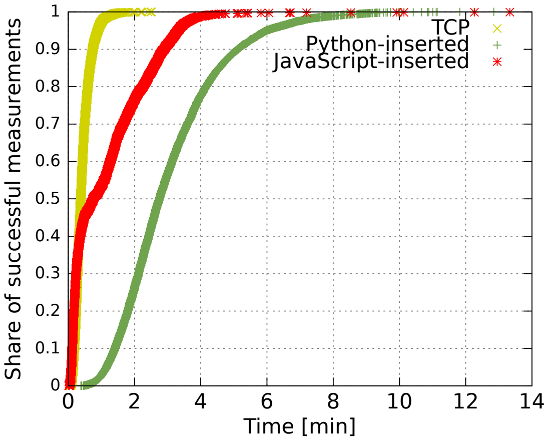
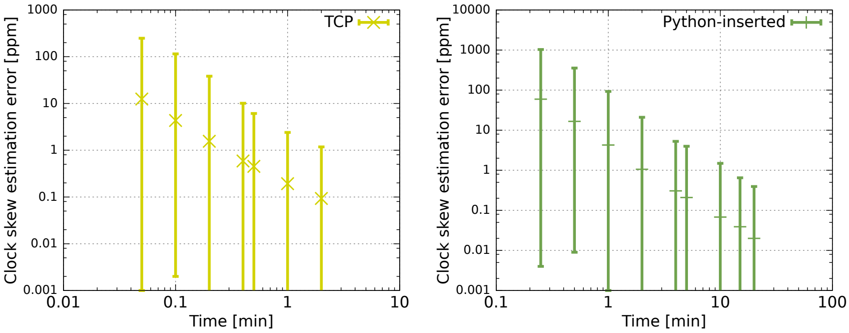


Figure 5: TCP-time-stamp-based clock skew fingerprinting took shorter time than fingerprinting based on Python-inserted time stamps.

The slow convergence of the Python-inserted time stamps is caused by a high volatility of ϵ. Even though the program received different time stamps after each call of the *time.time()* function, the call of the *time.time()* function results in the context switch from the kernel space to the user space after the time stamp is obtained in kernel space, then the Python interpreter might schedule garbage collector or another service routine during the construction of the HTTP request. Finally, the context is switched to the kernel space to send the HTTP request through the network stack. Since the context switching is not deterministic, the variance of ϵ increases and clock skew estimates converge to its correct value longer. JavaScript implementation in the Firefox browser seems to have lower variance of the ϵ which resulted in a very fast convergence of about 40 % of experiments with the convergence pace similar to the pace achievable from TCP time stamps. However, about 60 % of experiments took longer time to converge in comparison with TCP time stamp fingerprinting.

Figure 6 depicts the error during the first experimental scenario. The quality of the estimation improves in time, as expected by the equation 5.



a) TCP time stamps. b) Python-inserted time stamps.

Figure 6: Minimal, maximal and median value of the error during clock skew computation.

In summary, the observation confirms the expectations raised by the formula 5. The longer an experiment lasts, the less likely it is to get a time stamp that introduces an offset point that incorrectly shifts the upper bound of all offset points.

5 Imitating clock skew of a victim computer

One of the use cases introduced by previous research [Huang et al., 2012] employed clock-skew-based identification as a part of a multi-factor authentication.

In this scenario, a client authenticates to a server, for example, via Internet. The server quickly estimates the clock skew value. In case of stolen credentials, even though the provided primary authentication method, e.g. password, is correct, a newly observed clock skew triggers an additional authentication method, e.g. via SMS. Note that this scenario expects that it is hard or impossible to forge the clock skew of a computer; hence the scenario expects that an attacker cannot mimic clock skew of a computer that is regularly used to access the password-protected service.

When an NTP daemon [Mills et al., 2010] is running, it tracks the current correction for the built-in clock for the frequency of the local clock oscillator in a special file called *driftfile*. Consequently, when an NTP daemon is restarted, it can read the previously stored correction from the *driftfile*, modify the clock to compensate its ticks according to the correction, and use the value as a basis for future clock corrections.

Therefore, anyone trying to mimic a specific clock skew of another computer can follow the consequent steps:

1. Run an NTP daemon and find the clock skew of the attacking computer, e.g. from the *driftfile*.
2. Find the clock skew of the victim, preferably by fingerprinting the victim computer by the attacking computer (still running the NTP daemon).
3. Add the clock skew learnt in step 2 to the value stored in the *driftfile* in the step 1. Use the sum as the correction of attacking computer clock, for example, by restarting the NTP daemon on the attacking computer and immediately stopping the daemon.

By following the steps above, we were able to reproduce the built-in clock skew of the computers in our laboratory by another computer. As a result, the suitability of the clock-skew-based identification for the multi factor identification is questionable as it can be evaded easily.

A rapid fingerprinter cannot differentiate the attacker and victim computer. However, a long-term fingerprinter can observe differences in the clock skew development in time, e.g. related to temperature [Murdoch, 2006]. Consequently, the long-term fingerprinter can reveal that the attacking computer and the victim computer are indeed different devices.

6 Multiple IPv6 addresses

The advent of IPv6 brings the need to redesign the computer identification: in IPv4, every user used only one IP address at a single time. This enabled Internet sites to apply specific policies to a specific IP address. For example, a botnet computer trying to send a huge number of e-mails during a short time period can be blacklisted by the SMTP service.

A user that is connected via both IPv4 and IPv6 (dual stack) can connect to any dual stack service through IPv4 or IPv6. The addresses of two different families are not linked in any way. Therefore, the banned computer can evade the policies by switching the address family. Moreover, while IPv4 computers use only one address, IPv6-enabled computers can simultaneously use as many addresses as they can generate [Narten et al., 2007]. This Section demonstrates a novel use of clock skew estimates to link all IPv4 and IPv6 addresses of one computer.

Since the network layer does not modify data from upper layers, TCP time stamps and application-generated time stamps are present in IPv6 datagrams. However, ICMPv6 does not define ICMPv6 Time stamp Request and Reply.

6.1 Improved Linkability

The clock skew of a computer is not stable, temperature [Kohno et al., 2005, Murdoch, 2006] or time shifts [Polčák and Franková, 2014] influence its current value. Instead of estimating single stable clock skew, we propose to compute a sequence of clock skew values valid for a specific time frame. To improve the linkability of addresses of a computer, we consider two clock skew sequences to be similar, and consequently to be possibly of a single computer, if the following holds:

1. either both clock skew estimates were stable and both estimates were within the range of ±1 ppm (in this case, it does not make a difference if both addresses were active during the same period or not),

   or,

2. both clock skew estimates were within the range of ±1 ppm during the period when both addresses were active (in this case, there has to be a period during which both addresses were active; when clock skew of one of the addresses changed the other clock skew changed soon to a similar value).

The first case covers the original identification [Kohno et al., 2005]. Every computer with stable clock skew is identifiable even though it changes the IP address. The second case covers time shifts [Polčák and Franková, 2014] caused by NTP and user time manipulations. In principle the second case is similar to the evaluation used to detect hidden services [Murdoch, 2006].

6.2 Laboratory evaluation

To evaluate the correctness of our hypothesis that all addresses used by the same computer can be linked by matching their clock skew, we examined the data from the experiments described in subsection 4.2. During all fingerprinting measurements, the computers sent time stamps employing both IPv4 and IPv6. Both Python-inserted and JavaScript-inserted time stamps converged to the same value as the TCP time stamps in both IPv4 and IPv6.

In addition, we evaluated an Apple Mac Mini, an Apple Mac Book (Mac OS X 10.7.5 and earlier versions), an Apple iPad (iOS 8.3 and earlier versions), Windows 7, Windows 8.1, and FreeBSD computers. JavaScript-inserted time stamps converged to the same value for the same computer independently on the network protocol, i.e. IPv4 and IPv6. Except Apple computers, the estimated clock skew was within the ±1 ppm margin when computed from TCP time stamps. Moreover, the difference between the clock skew computed from TCP and JavaScript-inserted time stamps were also within the expected error level of ±1 ppm.

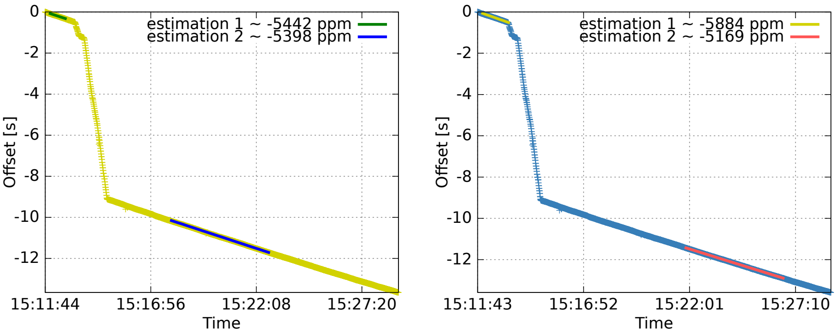
Clock skew of Apple devices estimated from TCP was completely different to the one computed from JavaScript-inserted time stamps. In our previous work [Polčák and Franková, 2014], we discussed exceptionally high clock skew of Apple devices when computed from TCP time stamps. This feature prevents the linkage of IPv4 and IPv6 addresses based on TCP time stamps. Although substantial changes of clock skew were detected in IPv4 and IPv6 packets at roughly comparable time, the estimated clock skew of IPv4 and IPv6 packets did not get to the expected error level. For example, the estimation computed from IPv4 datagrams of the Mac Book were between -5127 and -1900 ppm while estimated clock skew from IPv6 datagrams oscillated between -5032 and -1923 ppm. Similarly, the IPv4 clock skew estimations of the Mac Mini were in the range of -5152 and -1643 ppm while the IPv6 clock skew estimations varied between -5177 and -1037 ppm.

There are two system calls that influence clock values of a Linux, a FreeBSD, a Mac OS X, or an iOS computer:

- *adjtime()* slowly compensates the time difference so that the internal clocks are synchronized to the desired value, usually advertised by NTP, without a big jump of clock value. During this process, the system time exhibits different clock skew. After the adjustment is completed, the system time returns to the original clock skew value.
- *settimeofday()* replaces the local time at once without any influence on clock skew.

Our laboratory experiments linked all IPv4 and IPv6 addresses simultaneously used by a single computer even when the time stamps were influenced by the calls of *settimeofday()* and *adjtime()*. Clock skew estimations were stable and comparable in between the calls. Again, the only exceptions were estimations derived from TCP time stamps of Apple devices because of the exceptionally high and unstable clock skew estimates.

Although the addresses of the Apple devices were not linkable automatically, it is possible to link the addresses by thorough examination of the observed time stamps. Let us take Figure 7 as an example. It displays one fingerprintee that has both an IPv4 and an IPv6 address. Even though the estimated clock skew differs by hundreds of ppm, the shape formed by the observed offset points is similar. Hence, it is possible to detect changes in the estimated clock skew and correlate the addresses of the same computer even though the exact estimated clock skew value do not match. Nevertheless, to do so, the fingerprinter needs to have enough offset points, so that the clock skew changes are identifiable.



a) IPv4 address.

b) IPv6 address.

Figure 7: Clock skew measurement of an Apple device based on TCP time stamps.

7 Real network experiment

This section describes our experience from real network monitoring. For this purpose, we fingerprinted devices in our faculty network. We mirrored traffic going through the network link that connects our faculty with the University network. We fingerprinted TCP segments with time stamp options enabled. Our monitoring was completely passive and long term. For each detected address, we computed clock skew and followed its development. Originally [Polčák and Franková, 2014], we performed the experiment for IPv4 addresses. In this paper, we include IPv6 data for the study and look at the data from the perspective of a rapid and long-term fingerprinter.

We fingerprinted following devices:

- laboratory computers (Windows, Linux, and FreeBSD),
- desktops and laptops of the faculty staff (Windows, Linux, and FreeBSD),

- mobile devices connected to the faculty Wi-Fi (e.g. Android, iOS),
- faculty servers (mostly Linux and FreeBSD),
- remote servers accessed by the above mentioned computers,
- remote clients that connected to local servers.

Since 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 detail and we did not compare the gathered information to external sources. Table 2 reports the number of addresses observed during the experiment. The goal of this experiment was to gather more information about the feasibility of the clock-skew-based identification in real network environment.

	IPv4	IPv6
Working hours	350–649	100–196
Nights	120–170	15–40

Table 2: Number of IP addresses with computed clock skew during the real network monitoring.

From the perspective of a rapid fingerprinter, clock skew distribution is very important. 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 one sampled state of the network as an example — an afternoon of a working day.

At that point of time, 823 addresses sending TCP time stamps were detected in the network (646 IPv4 addresses and 177 IPv6 addresses). The histogram of all estimated clock skew in the network at that time, displayed in Figure 8, shows that substantial part of addresses have clock skew close to zero. The addresses with very high (over 1000 ppm) or low (less than −1000 ppm) clock skew showed similar properties of the Apple operating systems, e.g. the clock skew was not stable [Polčák and Franková, 2014].

The maximal number of IP addresses within the ±1 ppm range was 223 for IPv4 and 72 for IPv6. It means that 34.5 % of IPv4 addresses could have been considered to be assigned to the same computer. In the IPv6 case, the ratio of addresses possibly originating from the same computer is 40.7 %. We believe that this is caused by the time synchronisation through NTP.

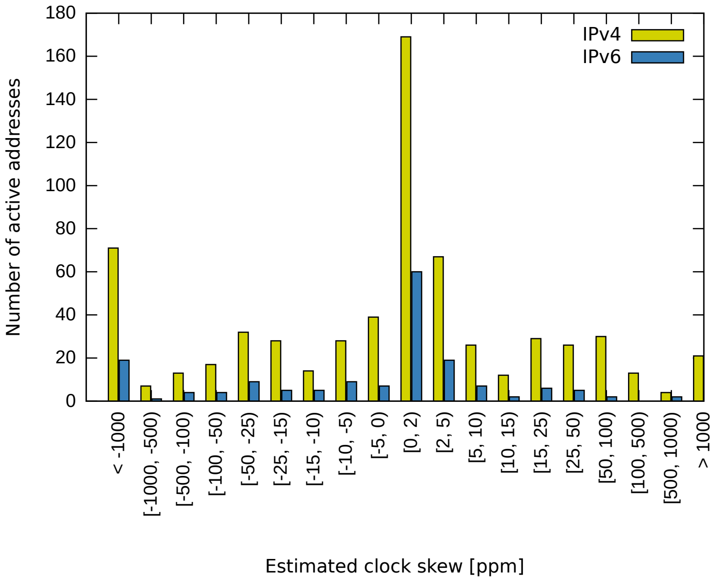
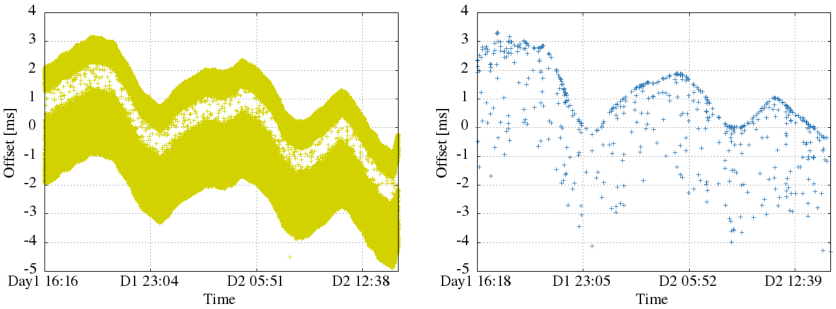


Figure 8: Histogram of clock skew distribution in real network. Note that the closer a bin is to zero ppm the smaller clock skew range it covers.

Similarly to the observation made by [Lanze et al., 2012], most of the estimated clock skew values are distributed between -100 ppm and 100 ppm, approximately 80 % of devices for both IPv4 and IPv6. As discussed in our previous work [Polčák and Franková, 2014], other researchers also detected that majority of devices have clock skew close to 0 ppm. Since clock manufacturers aim at producing clock as precise as possible, the fact that majority of devices have clock skew close to 0 ppm in an about 200 ppm range is not very surprising.

A long-term fingerprinter linking IPv4 and IPv6 addresses can detect changes in clock skew, e.g. made by NTP. Figure 9 displays an example of a computer running NTP. The computer employs both IPv4 and IPv6, however, the amount of the IPv4 traffic is much larger compared with the IPv6 traffic. Nevertheless, an experienced fingerprinter operator can detect similar changes around similar time. Hence a long-term fingerprinter can correlate the detected changes in the clock skew and link IPv4 and IPv6 addresses of the same computer.

The distribution of the observed clock skew of the real devices makes tracking a single computer in different locations hard, especially in large networks. Even a long-term fingerprinter cannot determine if the detected device is indeed the tracked device or a false positive, a device that happens to have a similar clock skew to the tracked device.



a) IPv4 address. b) IPv6 address.

Figure 9: Clock skew measurement of a Linux computer running NTP.

8 Discussion

As the previous research [Kohno et al., 2005,Murdoch, 2006,Polčák and Franková, 2014] highlighted, clock skew of a single computer is not completely stable. It is influenced by a number of physical factors, including temperature and voltage. Whenever a fingerprinter tries to compute the clock skew of a fingerprintee, the fingerprinter needs to account also for the other factors hindering the identification. These include:

- The operating system of the fingerprintee: the same computer have different clock skew in different operating systems [Kohno et al., 2005,Polčák and Franková, 2014] and the properties of time stamps vary, e.g. Windows clients do not initiate TCP sessions by default [Kohno et al., 2005] and Apple operating systems produce [Polčák and Franková, 2014] time stamps influenced by unstable, very high and volatile clock skew.

- Time adjustments: time changes on a fingerprintee are often propagated [Polčák and Franková, 2014] to the time stamp values. Time stamps added by user space applications are affected by both time synchronisation and manual changes of clock values. TCP time stamps are affected by NTP in Linux and FreeBSD; Mac OS X and iOS produce unstable and highly volatile TCP time stamps.

- The variable observation delay ϵ: increases whenever any intermediate network node experiences congestion. Additionally, depending on the time stamp source, the delay includes time stamp processing at both the fingerprintee and the fingerprinter. Moreover, the delay ϵ contains quantisation error, i.e. the time between observable clock ticks. Without any counter-measures, the quantisation error is higher for low frequency time sources.

A rapid fingerprinter suffers by all factors. In contrast, a long term fingerprinter can compare the shifts of different computers, e.g. to match the traits of different addresses of the same computer (see Figures 7 and 9 for examples). An active fingerprinter can remove [Zander and Murdoch, 2008] the quantisation error from ϵ by synchronized sampling.

Clock manufacturers try to make clock as precise as possible. Hence, the clock skews values are usually bound to be close to 0 ppm. The fingerprinting of our faculty network suggests that the clock skew of the majority of devices is in the range from -100 ppm to 100 ppm. This prevents a rapid fingerprinter to uniquely differentiate the devices. In contrast, a long-term fingerprinter can detect unique devices by observing changes in clock skew related to NTP, temperature etc.

A privacy seeking user may consider permanent synchronisation with precise time servers (which Apple devices do by default) to evade rapid fingerprinters. However, as NTP uses *adjtime()*, a small changes of clock skew triggered by the NTP daemon can help a long-term fingerprinter to detect unique computers. Nevertheless, frequent changes of clock value [Polčák and Franková, 2014] can prevent a fingerprinter to estimate the clock skew with required precision (see Figure 6). As 38.1 % of clock skew estimations from JavaScript-inserted converged in 20 seconds (during the experiments performed in Section 4), the changes need to be very frequent. Another option for a privacy seeking user is to deactivate TCP time stamps. As Windows clients do not initiate the inclusion of TCP time stamps, many TCP flows do not include TCP time stamps.

The clock skew imitation method presented in Section 5 can also be applied as a counter-measure for revealing computers behind network address translator or linking IPv6 addresses of the same computer. When a group of computers mimic the same clock skew value, the clock skew estimation results in a similar value for all computers. The fingerprinter is likely to evaluate all clock skew to belong to a single computer. Nevertheless, a long term fingerprinter can detect trails suggesting that more than one computer was detected.

As a result, rapid fingerprinting is more useful in small networks, preferably in networks where the fingerprinter can prevent time stamp manipulation and clock skew mimicking. A long-term fingerprinter can benefit from the identification of the precise moment and value of clock skew changes to link addresses of unique devices.

9 Conclusion

Clock-skew-based computer identification [Kohno et al., 2005] covers many use cases, such as deanonymization [Murdoch, 2006], fake wireless access point detection [Jana and Kasera, 2010], and web user identification [Huang et al., 2012]. The identification can be based on a detection of a stable clock skew that is unique for a single computer or by observing clock skew changes [Zander and Murdoch, 2008]. Based on the previous research, this paper identifies specific types of clock-skew-based fingerprinters and their requirements on clock skew estimation.

One of the main contributions of this paper is the study of 24.071 clock skew estimation samples. This study followed the findings of our previous paper [Polčák and Franková, 2014] and increased the knowledge of the requirements for getting sound clock skew estimates. This paper presents that the soundness of the clock skew estimation does not depend merely on the number of packets as previous studies suggested. Instead, the duration of the measurement is more important as it enables to consider offset points with bigger time span to form the upper bound (from which the clock skew estimation is derived). The bigger time span consequently allows a fingerprinter to compensate bigger variance in time stamp delivery time.

This paper covers a method allowing an attacker to mimic clock skew of another computer and mislead especially a rapid fingerprinter. Whenever an attacker can measure the clock skew of the victim computer, he or she can configure own computer to pretend to have the clock skew of the victim computer. The clock skew imitation method can also be used as a counter-measure from previously proposed clock-skew-based identification methods, such as reconnaissance attacks via clock skew estimation; a group of computers can be configured to display similar clock skew characteristics and thus confusing the fingeprinter into evaluating that only one computer is detected.

In addition, this paper studies the linkage of IPv4 and IPv6 addresses of the same computer using clock skew. The laboratory tests confirmed that the address family, i.e. IPv4 or IPv6, does not have any influence on a clock skew. Hence, it is possible to link IPv4 and IPv6 addresses of the same computer by estimating the clock skew of the device using each address. However, the real network study revealed the issues of real network clock-skew-based identification deployment as clock manufacturers typically aim at producing clocks with as small skew as possible.

One of the possibilities to improve the linkage of different IP addresses used by the same computer lays in the correlation of clock skew changes. When a fingerprinter has enough time stamps, he or she can correlate the IP addresses by observing the same change in the clock skew or clock value in all clock skew estimations computed for the addresses of the same computer.

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.

We would like to thank the anonymous reviewers of this paper and the anonymous reviewers of our previous work [Polčák and Franková, 2014], their comments improved our research.

References

[Borman et al., 2014] Borman, D., Braden, B., Jacobson, V., and Scheffenegger, R. (2014). *TCP Extensions for High Performance.* IETF. RFC 7323 (Proposed Standard).

[Huang et al., 2012] Huang, D.-J., Yang, K.-T., Ni, C.-C., Teng, W.-C., Hsiang, T.-R., and Lee, Y.-J. (2012). Clock skew based client device identification in cloud environments. In *Advanced Information Networking and Applications*, pages 526–533.

[Jacobson et al., 1992] Jacobson, V., Braden, B., and Borman, D. (1992). *TCP Extensions for High Performance.* IETF. RFC 1323 (Obsoloted by RFC 7323).

[Jana and Kasera, 2010] 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 et al., 2005] Kohno, T., Broido, A., and Claffy, K. (2005). Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2):93–108.

[Lanze et al., 2012] 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 et al., 2010] Mills, D. L., Martin, J., Burbank, J., and Kasch, W. (2010). *Network Time Protocol Version 4: Protocol and Algorithms Specification.* IETF. RFC 5905 (Proposed Standard).

[Murdoch, 2006] 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.

[Narten et al., 2007] Narten, T., Draves, R., and Krishnan, S. (2007). *Privacy Extensions for Stateless Address Autoconfiguration in IPv6.* IETF. RFC 4941 (Draft Standard).

[Papagiannaki et al., 2002] Papagiannaki, K., Moon, S., Fraleigh, C., Thiran, P., Tobagi, F., and Diot, C. (2002). Analysis of measured single-hop delay from an operational backbone network. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 535–544.

[Polčák and Franková, 2014] Polčák, L. and Franková, B. (2014). On reliability of clock-skew-based remote computer identification. In *International Conference on Security and Cryptography.* SciTePress - Science and Technology Publications.

[Polčák et al., 2014] Polčák, L., Jirásek, J., and Matoušek, P. (2014). Comments on "Remote physical device fingerprinting". *IEEE Transactions on Dependable and Secure Computing*, 11(5):494–496.

[Sharma et al., 2012] 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 and Murdoch, 2008] 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.