Clock Skew Based Remote Device Fingerprinting Demystified

Fabian Lanze, Andriy Panchenko, Benjamin Braatz, Andreas Zinnen
{firstname.lastname}@uni.lu
Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg

***Abstract*—Commonly used identifiers for IEEE 802.11 access points (APs), such as network name (SSID), MAC, or IP address can be easily spoofed. This allows an attacker to fake a real AP and intercept, collect, or alter (potentially even encrypted) data.**

In this paper, we address the aforementioned problem by studying limits of unique remote physical device identification based on their clock skew—an unavoidable phenomenon that causes clocks to run at marginal but measurably different speed. To this end, we propose an algorithm for passive fingerprinting using timestamps regularly sent by APs in beacon frames. The major advantages of our method are that it is online and that we are able to eliminate the influence of clock skew of the measurement device. Hence, fingerprints performed by different devices become comparable. We calculate the precision of our clock skew measurement algorithm and provide a termination criterion for estimation of the clock skew with arbitrary precision. Moreover, conducting a large scale evaluation, we study the stability and uniqueness of clock skew as a means for remote wireless device identification.

I. Introduction

Up to recently, Internet access was mostly performed from the same "trusted" places (e. g., home, office), which imposed no particular challenges on trust in service providers. Today, mobile devices offer the possibility to be connected all the time as public access points (APs) become omnipresent. We are more and more in situations where we connect to "unknown" networks because of cost-efficiency, e. g., in the shopping mall, in coffeehouses, on the street. Since commonly used identifiers, such as network name (SSID), MAC, or IP address can be easily spoofed, malicious access points can pretend to be those known to the user and act as man-in-the-middle, i. e., intercept, collect for further misuse, or even alter the data. Moreover, association with a malicious access point enables various attacks on connected devices as the latter can be directly addressed.

The arguments as listed above raise the necessity for methods and tools that enable remote device identification. A straightforward solution to this problem is the usage of Robust Security Network Association in 802.11i standard. These authentication protocols apply cryptographic primitives using an additional authentication server [18], which requires setup and maintenance. Since entities running open access points usually do not have an incentive for such an additional measure, the approach is not widely used.

Therefore, there is need for a solution that is able to provide reliable identification of hotspots, i. e., *device fingerprinting*, without any active cooperation of the hotspot provider. The focus of our work is on a method that is promising to yield unique identification of devices (even of the same vendor and series) and that can be performed by ordinary clients without any special hardware.

In this paper, we study limits of unique remote physical device identification based on their *clock skew*, a physical trait leading to tiny yet observable differences in clock speed. In all regular electronic clocks (and, thus, the ones in access points) the clock signal is produced by a *crystal oscillator*. The frequency of this oscillator depends on manufacturing parameters (e. g., the cut angle) and the type of crystal used. The imperfection in mechanic accuracy during the production process leads to slightly different frequencies even for crystals of the same type, series, and production date [9]. The clock skew is commonly given by a measure of *parts per million (ppm)*. Since clock skew is based on hardware characteristics, it promises to provide a reliable device identification that cannot be easily manipulated.

In our approach, we use timestamps regularly sent by access points in *beacon frames*. These are management frames that are periodically transmitted by access points to announce the presence of a wireless network. Beacons include, among others, timing information in microseconds. This information is used by a timing synchronization function (TSF) that keeps the sending and receiving time slots synchronized for all stations in a basic service set (BSS)—the AP with all associated clients. Each client adopts the timing broadcasted by the access point. The TSF timestamps in beacon frames are sent at a high frequency (typically every 100 ms), are of high precision, and, most notably, do not experience processing delays before sending (by specification).

All currently existing approaches in the field of clock skew based wireless device fingerprinting require modification of the device driver. Moreover, they fail to eliminate the clock skew of the measurement device and, hence, they are not able to provide a method for comparable remote device fingerprinting when using different measurement devices. Additionally, all of the proposed methods were only tested on a small proof-of-concept set of APs and measurement devices. Therefore, no conclusions on the uniqueness of devices' clock skews can be drawn. With our methods and evaluations we claim to overcome these drawbacks and demystify the addressed topic along several dimensions.

Contribution: Our contributions are as follows: (i) We propose a *mathematical model* for describing the influence of

the measurement device's clock skew on the estimation of a remote clock skew. Based on this model, we propose a method that estimates and removes the clock skew of the measurement device, thus, *making results from different fingerprinters comparable*. (ii) We provide an efficient *online algorithm* with *arbitrary precision*. Our method neither requires modification of the kernel nor of the driver and can be performed on most out-of-the-box Linux/Unix systems. (iii) We provide the most *comprehensive evaluation* of remote clock skew estimation and expose how distinct different physical devices are based on the information leakage from TSF time of WiFi chipsets. We come to the conclusion that TSF information can be an indicator to differentiate between different physical devices. However, it is not as unique as thought before (e. g., [11]) to reliably identify single devices.

II. Related Work

In general, the methods for *remote physical device fingerprinting* can be classified into *active* and *passive*. While active techniques require an interaction with the device to be fingerprinted (the *fingerprintee*), a mere observation of traffic is sufficient for passive methods.

Active methods are proposed by Sieka [20] and Bratus et al. [3]. Sieka uses precise measurements of the time it takes for a node to perform steps of the authentication procedure. The approach requires two different measuring devices for fingerprinting. Bratus et al. propose to use active behavioral fingerprinting based on malformed stimuli response, i. e., how devices react on non-standard and malformed 802.11 frames. In general, such active methods are detectable by the fingerprintee and can even potentially cause a disruption of the regular network communication.

Passive methods do neither require any cooperation with other nodes nor can be detected by the fingerprintee. Due to that reason, we focus on passive techniques.

It is possible to optimize the fingerprinting exactness if the deployed solutions are not limited to software. Accuracy rates of more than 99 % are shown to be possible investigating wireless frames in the modulation domain [4] or using a technique called *radio frequency fingerprinting* (RFF) [21], [19], [8], [7]. Nevertheless, such fingerprinting methods need dedicated hardware to identify the physical properties of the radio signal. Thus, they are not suitable in our context, since we want to fingerprint from standard mobile clients. Other passive approaches on remote fingerprinting focus on identifying unique device types [6] or device driver types [5]. Hence, those methods do not satisfy the requirement to differentiate between APs with the same hardware and firmware.

Bahl et al. [2] propose a method for identifying faked APs by analyzing the sequence number in the headers of 802.11 frames. These numbers are supposed to intermix if two APs advertise the same BSSID. Hence, it is only possible to detect the presence of two APs with the same BSSID but not to tell which of them is faked. Moreover, this approach is only applicable if both APs are active simultaneously in a nearby location.

Another class of approaches is utilizing the phenomenon called *clock skew*. Based on work of Moon et al. [15], Kohno et al. [12] introduced the concept of clock skew based remote device fingerprinting using the TCP Timestamps Option in TCP headers (which, when enabled, contain a 32-bit timestamp generated by the creator of a packet, see RFC 1323 [17]) having timing resolution in milliseconds. The clock skew is estimated with linear programming (LP). Clock skews are shown to be distinguishable among different physical machines yet stable over time. Still, the approach requires observing a TCP connection to the fingerprintee while wireless APs can not be directly accessed via TCP in general.

Clock skew fingerprinting in a wireless scenario based on the TSF timestamps in beacon frames was first studied by Jana et al. [11]. Instead of the LP method the authors use a least squares fit estimation (LSF), which is more sensitive to outliers, as they expect less outliers compared with timestamps in TCP packets. Moreover, using TSF data requires significantly smaller sample sizes due to higher clock resolution. For measuring the receiving time of a beacon frame, a modified driver is used. The authors argue that it is not possible to fake the clock skew using only software because of unpredictable sending delays due to Medium Access Control. Still, the fingerprints are not comparable between different machines due to the influence of the fingerprinting device's own clock skew.

The possibility to get access to a precise clock source was further addressed in [1]. Arackaparambil et al. utilize the TSF timer; however, their technique does not remove the skew of the fingerprinter card from the clock skew estimation and is only evaluated with two WiFi chipsets of the same type.

In general, all attempts for wireless device fingerprinting using clock skew proposed so far depend on modified drivers and lack a large-scale evaluation. Moreover, none of the described methods is able to eliminate the effect of the fingerprinting device's own clock skew from the estimation. Hence, the estimation is not comparable between different devices. To the best of our knowledge, our proposed method is the first that addresses and solves all challenges mentioned above.

III. Measuring Clock Skew

In this section, we introduce our approach for estimating the clock skew of wireless access points. We start by giving a mathematical model of clocks and clock skews and extrapolate how the skew difference between a measuring device and a measured device can be computed from timestamps taken from both of them. Then, we explain in detail how we obtain these timestamps in practice, how we remove the fingerprinter's own skew from the result and how we approximate the skew from noisy measurement data.

A. Mathematics of Clock Skews

A *clock* c counts time steps from some initial point in true time i_c, where all clocks in this paper use microsecond resolution. This initial point in time depends on the concrete device. For example, the main system clock in Unix operating

systems uses 1970-01-01 00:00 as the initial point. Other clocks may use system start-up or even leave the initial point unspecified.

The timestamp reported by a clock c at a point t in true time is denoted by $\mathrm{R}_c(t)$. If we had access to an absolutely exact clock ex the reported time would be $\mathrm{R}_{ex}(t) = t - \mathrm{i}_{ex}$. The clocks used in standard hardware are, however, not high-precision atomic clocks and, hence, a clock c has an *offset* $\mathrm{off}_c(t) = \mathrm{R}_c(t) - (t - \mathrm{i}_c)$, whose absolute value generally increases over time.

The *skew* s_c of a clock c is the first derivative of its offset, i. e., the slope of the difference between measured time and true time. A positive skew means that the clock is too fast, while a negative skew shows that it is too slow.

We assume the skew to be constant (and, therefore, the offsets to develop linearly) for the duration of a skew measurement. This assumption is justified by conclusions from related work as well as our own experiments presented later in this paper. Hence, the skew in a reasonably small time window between t_1 and t_2 can be computed by

$$\mathrm{s}_c[t_1, t_2] = \frac{\mathrm{off}_c(t_2) - \mathrm{off}_c(t_1)}{t_2 - t_1}$$

or, equivalently, by

$$\begin{aligned}\mathrm{s}_c[t_1, t_2] &= \frac{(\mathrm{R}_c(t_2) - (t_2 - \mathrm{i}_c)) - (\mathrm{R}_c(t_1) - (t_1 - \mathrm{i}_c))}{t_2 - t_1} \\ &= \frac{\mathrm{R}_c(t_2) - \mathrm{R}_c(t_1)}{t_2 - t_1} - 1 \ .\end{aligned}$$

However, we typically do not have access to an absolutely exact clock. Hence, we have to take the skew of the measuring device's own clock m into account.

Rewriting the equation above, we have

$$\mathrm{R}_c(t_2) - \mathrm{R}_c(t_1) = (1 + \mathrm{s}_c[t_1, t_2]) \cdot (t_2 - t_1) \qquad (1)$$

for the measured clock (in our case, the AP's TSF clock) and

$$\mathrm{R}_m(t_2) - \mathrm{R}_m(t_1) = (1 + \mathrm{s}_m[t_1, t_2]) \cdot (t_2 - t_1) \qquad (2)$$

for the clock of the measuring device (the fingerprinter's system clock). If we now measure the time between t_1 and t_2 with both clocks then we can compute (by substituting Equations 1 and 2 and canceling $t_2 - t_1$)

$$\begin{aligned}&\frac{(\mathrm{R}_c(t_2) - \mathrm{R}_c(t_1)) - (\mathrm{R}_m(t_2) - \mathrm{R}_m(t_1))}{\mathrm{R}_m(t_2) - \mathrm{R}_m(t_1)} \\ &= \frac{\mathrm{s}_c[t_1, t_2] - \mathrm{s}_m[t_1, t_2]}{1 + \mathrm{s}_m[t_1, t_2]} \approx \mathrm{s}_c[t_1, t_2] - \mathrm{s}_m[t_1, t_2] \ .\end{aligned}$$

The denominator can be neglected in the last expression, since observed clock skews are smaller than 100 ppm, i. e., $\mathrm{s}_m[t_1, t_2] \ll 1$ and, hence, the error due to neglecting it is relevant only in the fifth significant digit.

In summary, we can compute the difference between the clock skew of a measured device and the clock skew of a measuring device from timestamps taken by both of them. In order to be usable as a fingerprint of the measured device, which has to be comparable between different fingerprinter devices, we need an estimation of the fingerprinter's own skew s_m and then add it to the resulting expression above to obtain the pure skew s_c of the fingerprintee.

In the following sections, we discuss how we use this in practice. In Sect. III-B, our method for taking the timestamps R_c from an access point as well as R_m from the measuring wireless client is explained. In Sect. III-C, we propose to use the drift recorded by NTP implementations as an estimation of the measuring device's skew s_m. Finally, in Sect. III-D, we illustrate how we approximate the clock skew as a linear slope, because in practice we do not only measure two points in time, but a sufficiently large sample in order to counteract possible measuring inaccuracies.

B. Determination of Timestamps

In order to measure the clock skew of an access point's TSF clock, two timestamps, namely R_c for the fingerprintee and R_m for the fingerprinter, have to be taken at multiple points in time. In this case, R_c is the current value of the access point's TSF timer when a beacon frame is sent and R_m reflects the time of the fingerprinter's clock when the same beacon is received.

The current value of the access point's TSF timer is available in an almost optimal manner. Beacon frames contain a 64-bit timestamp in microsecond resolution, which equals—according to 802.11 specification—the value of the TSF timer at the time where the data symbol containing the first bit of the timestamp is tramsmitted (plus the delay caused by its physical interface before the actual transmission). Thus, assuming an appropriate implementation, this timestamp is perfectly suitable for clock skew estimation and requires no further improvements.

A determination of the fingerprinter's receiving time is more challenging. The estimation's accuracy strongly depends on the clock source. Precise measurements have to be performed in kernel space in order to avoid effects caused by unpredictable processing delays. Previous works modified drivers to overcome this problem. In [11], an adapted driver executes a `dogettimeofday()` system call when a packet is received. Hence, the critical measurement is shifted from user space to kernel space. The driver modifications presented in [1] are necessary to utilize the client's TSF timer as clock source.

Our approach avoids a modification of the used drivers. We implemented a lightweight tool for accurate recording of beacon frames based on the Python library *scapy*[1]. For measuring the receiving time, it performs an ioctl call with the request code `SIOCGSTAMP`. On Unix-based systems, ioctl (short for input/output control) calls are dispatched by the kernel to a driver in order to perform device specific operations. In our case, we obtain a timestamp that is recorded by the driver when receiving the frame by querying the kernel for its system clock value in microsecond resolution. Hence, we are able to achieve the same accuracy as in [11] without altering any driver implementation. We tested this method on

[1] http://www.secdev.org/projects/scapy/

several customary laptops running different Linux distributions and found the ioctl call to be implemented and function correctly for each wireless NIC driver that supports monitor mode.

C. Elimination of Measuring Device Skew

Once the sample of timestamps is recorded, there still remains one challenge, which could not be solved so far. As shown in the mathematical model, we can only compute $\mathrm{s}_c - \mathrm{s}_m$, i. e., the *subjective* skew between the access point's TSF timer and the fingerprinter's system clock. To use the skew as fingerprint, this influence has to be removed to make measurements performed by different fingerprinters comparable. We propose to use the drift recorded by an NTP daemon as an estimation of s_m.

In general, NTP is designed to synchronize clocks over network connections. For details of the specification, we refer to [13]. On Unix-based systems, *ntpd* is a daemon process that implements a *software phase-locked loop*. Based on a complex clock discipline algorithm, it calculates the clock offset using a phase/frequency predictor. For further details about the exact procedure, we refer to [14]. Finally, this offset is used to calculate a correction for the system clock which is applied by the kernel once each tick interrupt with the aim of minimizing the clock skew.

Nevertheless, it is not necessary to run *ntpd* during the recording of beacon frames. *ntpd* stores the clock drift it has observed over time as a constant in a file in order to reuse this value, e. g., for re-initialization after reboot. Therefore, it is enough to calculate this drift only once. We read the clock drift value and use it as approximation of s_m. By adding this value to the subjective clock skew estimation, we obtain a precise estimation of the access point's clock skew s_c, which is now fingerprinter independent. In preliminary experiments we found that after running *ntpd* for some hours the accuracy of this constant value is sufficient.

D. Approximation of Slope

The last remaining step for estimating clock skew as device fingerprint is how to actually approximate the slope of the offset pattern. We use a data set consisting of n points (data pairs) (x_i, y_i), $i = 1, \ldots, n$, where $x_i = \mathrm{R}_m(t_i) - \mathrm{R}_m(t_1)$ is the receiving time with respect to the fingerprinter's clock and $y_i = (\mathrm{R}_c(t_i) - \mathrm{R}_c(t_1)) - (\mathrm{R}_m(t_i) - \mathrm{R}_m(t_1))$ is the offset between the TSF time and the receiving time. Due to the before mentioned assumptions, we expect this sample to follow a linear pattern. Simple *least squares fit* (LSF) linear regression is a standard approach to estimate two parameters α and s for which the straight line $y = \alpha + s \cdot x$ fits best into the observed data. The solution minimizes the sum of the squares of the estimation errors. In the following, we show how to efficiently solve this problem in order to calculate the slope s as approximation of $\mathrm{s}_c[t_1, t_n] - \mathrm{s}_m[t_1, t_n]$.

LSF as well as the *margin of error* (i. e., the radius around the current slope estimation containing the true slope with a confidence of 95 %) for the resulting approximation are usually computed with respect to all data points observed. Since stabilization of the approximation is not predictable a priori due to noisy data, we would need to compute both values—the slope estimation and the margin of error—over the whole data set at each newly received beacon to determine if the margin of error is sufficiently small to terminate. This is not feasible on fingerprinters with limited resources—such as mobile devices. Therefore, we propose to use an online variant of LSF and a heuristic for termination, both of which can be incrementally computed in constant time for each new beacon.

Using the displacement law (Steiner theorem) the slope estimation in the n-th step can be expressed as

$$s(n) = \frac{(\sum_{i=1}^{n}(x_i y_i)) - n \cdot \bar{x}_n \bar{y}_n}{(\sum_{i=1}^{n} x_i^2) - n \cdot \bar{x}_n^2} \quad .$$

In order to calculate $s(n+1)$ all sums that have to be calculated can be split into a sum, that has already been built in the previous step and an addend that covers the new value (x_{n+1}, y_{n+1}). Hence, it is sufficient to only store the values $\sum_{i=1}^{n} x_i y_i$, $\sum_{i=1}^{n} x_i^2$, $\sum_{i=1}^{n} x_i$ and $\sum_{i=1}^{n} y_i$ from the previous iteration in order to adjust the LSF estimation for the new values, i. e., the receiving time and offset for the next beacon frame.

An indicator for the accuracy that does not require knowledge of all data points is the evolution of slope estimations. As the online LSF calculation stabilizes, the difference of succeeding estimations will decrease since the influence of new points declines. We define two values: A *threshold* θ, which expresses the maximum acceptable difference of two succeeding estimations and a *threshold counter* c_θ. Let $\mathrm{LSF}_{\mathrm{onl}}(n)$ be the estimation of the slope using online LSF after the n-th iteration. Then, in each iteration n, we calculate

$$\Delta(n) = |\mathrm{LSF}_{\mathrm{onl}}(n) - \mathrm{LSF}_{\mathrm{onl}}(n-1)| \quad .$$

The algorithm terminates, when $\Delta(n) < \theta$ in c_θ succeeding iterations. The parameters θ and c_θ can be used to define a trade-off between the minimum level of accuracy required by the application and the duration of estimation.

In Sect. IV-B we show how to practically determine the parameters and evaluate our approach.

IV. Evaluation

A major contribution of this paper is a comprehensive evaluation of remote clock skew estimation. In particular, we managed to fingerprint 388 physical access points. In this section we provide the results of our evaluation and expose how different physical devices can be distinguished based on the information leakage from the TSF time in beacon frames. At first we describe how we collect the data. We then propose and evaluate a method to determine the parameters for the termination heuristic proposed in Sect. III-D. Finally, we reveal our study on two most important properties of clock skews: their distribution between different devices (from which one can conclude its uniqueness) and its stability over time (that is a means for its applicability).

A. Data Set

In this section we describe how we collected the data for evaluation. In order to obtain an extensive and independent data set, we measure the clock skew of as many different APs as possible by applying the following procedure.

To make the clock skew estimation independent of the fingerprinting device, we determine the values of s_m by running the NTP daemon for 48 hours on each laptop. This is done only once at some point of time prior to running the experiments.

Our method probes for used 802.11 channels and stores the TSF timestamps extracted from beacon frames together with the receiving timestamps as described in Sect. III-B over a period of 10 minutes for each channel that has at least one active AP. Thus, up to 6,000 beacons can be recorded for an AP assuming standard configuration of the inter-beacon interval (i. e., 100ms).

During our experiments we used 5 different laptops running Ubuntu 10.10 and the standard driver provided by the distribution for the appropriate WiFi chipset. Note that all these laptops are equipped with a different WiFi chipset and, hence, all used different drivers (see Table I for details).

	Wifi chipset	**Driver**
reddog	Intel Pro/Wireless 2200BG	ipw2200
SERRES	Broadcom BCM4312 802.11a/b/g	b43
RDIntel	Intel Pro/Wireless 3945ABG	iwl3945
XAMES	Intel Pro/Wireless 4965	iwlagn
Lumpi	Atheros AR5001X+ PCMCIA	ath5k

TABLE I
WIFI EQUIPMENT ON THE FINGERPRINTER HOSTS

We ran the experiment at 63 different places, including residential areas, public places (e. g., city center), and university campuses in 4 different countries (Germany, Luxembourg, France, and Belgium). Samples with less than 500 beacons over 10 minutes (which occur, e. g., due to a poor signal level) were discarded. This resulted in a total number of 388 measured physically different APs (we excluded virtual multi-SSID networks that operate on the same AP as the same TSF timer is used for their beacon generation and, thus, they have the same clock skew). Consequently, we were able to build a data set containing approximately ten times as much APs as in the biggest evaluation performed so far [11]. While Arackaparambil et al. [1] evaluated their approach using only 2 APs of the same vendor and the same chipset, our data set contains data from more than 50 different vendors.

The data was captured on live systems with real user traffic. In preliminary experiments we verified that traffic on the wireless medium has no significant influence on our approach.

In summary, our data set is notably representable and independent as it contains data from a large number of different devices, from different places, times, types of access points (public/private) and the captured APs show wide distribution across different vendors. To the best of our knowledge, we collected the most comprehensive data set in the area of clock skew measurements with "real world" APs so far.

B. Defining Termination Parameters

In Sect. III-D, we have shown how to apply parameters θ and c_θ in order to determine when the online algorithm for clock skew estimation should terminate. Here, we describe a method how to define suitable values for these parameters.

We first evaluate the average margin of error as a function of beacon frame sample size for a training set of 200 APs in our dataset, shown in Figure 1. The plotted margin of error is the range of values above and below the estimated clock skew, i. e., our sample statistic. The reference value at each point is the estimated value of the clock skew based on the data seen up to this point. The margin of error corresponds to a 95 % confidence interval. Taking the plot as a reference, we can find the number i of beacon frames when the required precision is reached. We propose to observe the absolute values of changes in the estimation slope in the last steps before the i-th, i. e., $|\mathrm{LSF}_{\mathrm{onl}}(i) - \mathrm{LSF}_{\mathrm{onl}}(i-1)|$, $|\mathrm{LSF}_{\mathrm{onl}}(i-1) - \mathrm{LSF}_{\mathrm{onl}}(i-2)|$, etc. This way, we are able to find values of θ and c_θ that lead to the approximation of the desired accuracy.

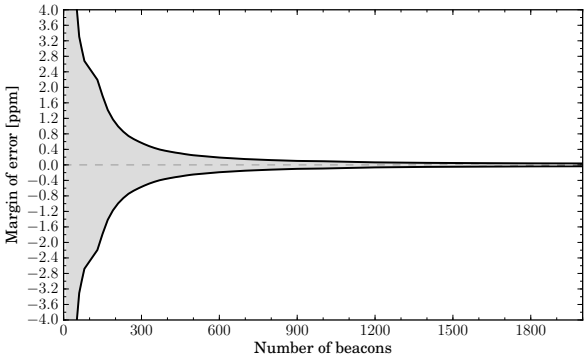

Fig. 1. Average margin of error using online LSF as function of beacon sample size

To evaluate our termination heuristic, we fixed two margins of error and determined corresponding pairs of θ and c_θ: one of them is more precise, the other is less precise yet much faster in clock skew estimation. We selected the margins of error to be 0.15 ppm for the first and 0.5 ppm for the second. This precision is reached on average after approximately 800 and 300 beacon frames, respectively. This corresponds to 80 and 30 seconds of listening time.

Figure 2 shows the values of θ and c_θ, derived as described above, together with the corresponding cumulative distribution function (CDF) for the resulting margin of error when applying our heuristic on the test set of 188 remaining access points (disjoint from the 200 APs that were used for learning the parameters). The results show that indeed in 88 % of all cases the estimation is at least as precise as 0.15 ppm (0.5 ppm for the second scenario). Moreover, 65 % of all estimations

are at least as precise as 0.05 ppm (0.15 ppm in the second scenario). Therefore, we can conclude that our termination heuristic provides a precise estimate for the accuracy defined beforehand.

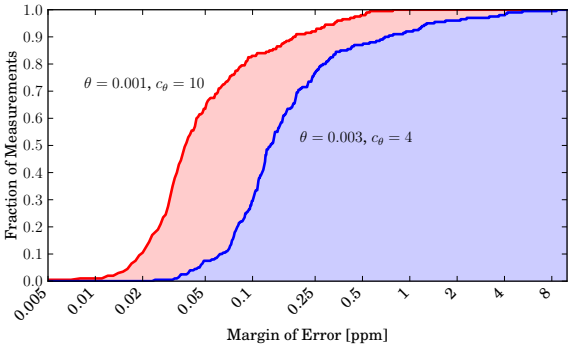

Fig. 2. Cumulative distribution function of the margins of error for different values of θ and c_θ

C. Distribution of Clock Skews

It is important to know how unique the clock skew of a device is. Even very precise algorithms for the estimation will not help to uniquely identify APs if many devices have the same value of the clock skew. In this section, we demystify the clock skew as a measure for unique device fingerprinting.

The distribution of clock skews in our data set is shown in Fig. 3. It reveals that all clock skews are in a rather short range between -30 ppm and 30 ppm with a bias towards negative values. This is a contradiction to the belief that there is a significant difference in TSF clock skews of wireless access points. Nevertheless, the trend of this distribution can also be found in the results of Jana et al. [11]. Recall that the authors evaluated only 41 access points and were not able to make results measured by different fingerprinters comparable. The maximum difference of two AP's clock skews in each of their traces lies also in magnitudes of 30–40 ppm (even though, due to the presence of the fingerprinter's skew in their measurements, their values are in a different range). In their results one very distinct outlier occurs with an estimated clock skew of less than -1000 ppm. During our experiments such results also appeared infrequently. After investigating the data in detail we found that such estimations were always a result of the AP's TSF counter being reset to zero during the measurement.

The reason for the observed distribution of clock skews to be within the range of ± 30 ppm is probably due to quality specification constraints by vendors. It is to be expected that for chrystal oscillators which are used for critical operations such as the 802.11 communication protocols, a preselection is done by manufacturers to assure a minimum level of quality. This is also consistent with the recommendation in the 802.11 standard [10].

Our evaluations show that the clock skew alone cannot serve as a unique fingerprint for wireless access points. The observed distribution mitigates the previously believed assumption. However, clock skew can still be used to differentiate between the APs. It is also of vital importance to study the variation of clock skew estimations over time and altering external conditions, such as the temperature. We investigate this variation and present the results in the following section.

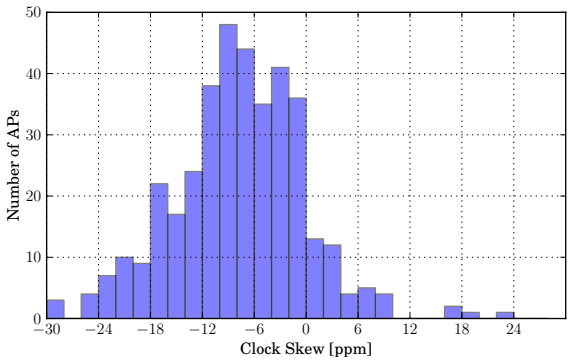

Fig. 3. Distribution of clock skews in the data set

D. Variation of Clock Skew

With the goal of using clock skew as fingerprint it is important to define a range of tolerable estimation values for recognizing an AP. It was already shown before that external factors, e. g., temperature differences, can influence the clock skew in the order of 1–2 ppm [12], [16]. Taking into account that clock skew can only be approximated, it is to find out if the variation range of different clock skew measurements for a single AP is determined by the accuracy of the approximation method or by changes of the environmental conditions.

Figure 4 shows two measurements of a single AP performed by the same fingerprinter over different hours of one day. It illustrates that the difference in clock skew estimation is no result of inaccuracy of the underlying estimation algorithm LSF since both samples are characterized by clearly distinguishable slopes. Consequently, since the accuracy of approximation with LSF is much more precise (± 0.005 ppm on average after 10 minutes of sniffing) than this exemplary variation of the clock skew (0.86ppm), we argue that the precision of LSF is sufficient and, instead, it is important to investigate the influence of environmental conditions on clock skew.

Therefore, we conducted an experiment where we measured the skew of four different APs for all 30 minute intervals over a period of one week by four different fingerprinters. The progress of temperature characterizes a regular week in an office setting leading to a temperature variation within approximately 5 °C during the experiment. Figure 5 shows the distribution of estimation values across all fingerprinters, visualized with boxplots. The clock skew estimation varies significantly in a range of 1 ppm over the period of one week even for this rather stable indoor setting. All used fingerprinters exhibit similar spans, where the measured variations can be explained by fluctuations of the APs' as well as the fingerprinters' clock skews (e. g., temperature changes may be

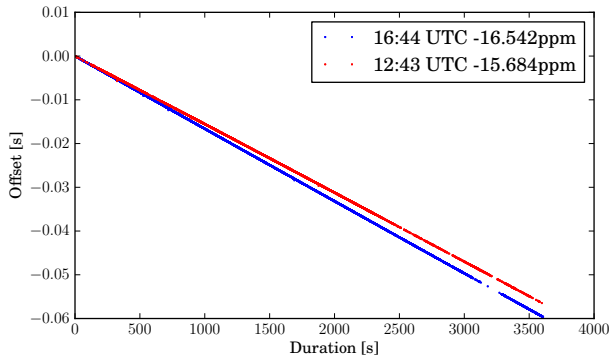

Fig. 4. Two measurements of the same AP by the same fingerprinter at different times

compensated differently by fan control systems of fingerprinter devices).

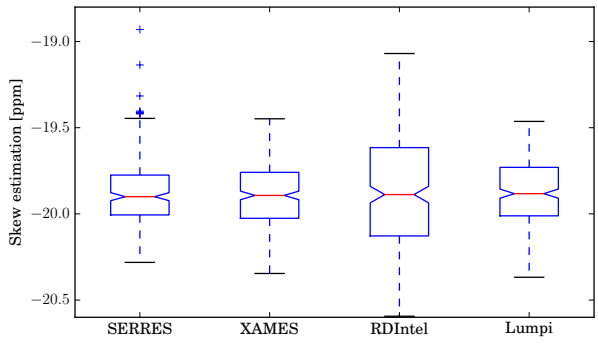

Fig. 5. Variation of clock skew measurements

This experiment further helps to verify the *ntpd* correction value as valid approximation of s_m. From the visualization we can deduce that the medians of all distributions are very close and since the notches of all boxes overlap we can conclude with 95 % confidence that the true medians do not differ significantly. Hence, the clock skew estimations are comparable between all four fingerprinters and, therefore, the approximation of s_m holds true in practice.

To conclude, the influence of environmental conditions for clock skew estimation seems to be unpredictable in practical usage and a clock skew difference of 1 ppm has to be accepted for recognition of a clock skew based fingerprint.

V. Conclusion

In this paper, we have studied the limits of wireless device identification based on clock skew. We have shown how to estimate the clock skew without any modification of the driver or operating system. We utilize timestamps regularly sent by access points in beacon frames—which is, to the best of our knowledge, the most precise remote time information that is disclosed. The major advantages of our method are that it is online, able to achieve arbitrary precision, and eliminates the influence of the measurement device's skew. Therewith, fingerprints performed by different devices become comparable. We conducted a large scale evaluation to explore the distribution and stability of clock skew among different access points and measurement devices. We found that the fluctuation of clock skew reaches up to 1 ppm. Moreover, a significant number of devices share a similar skew value. Hence, even though the clock skew restricts the set of possible devices, it cannot serve as a unique fingerprint for a wireless access point and has to be enriched with other features to achieve uniqueness.

Acknowledgements

This work has been supported by the National Research Fund of Luxembourg (FNR) within the AFR grant, CORE project MOVE, and EU FP7 PPP project OUTSMART.

References

[1] C. Arackaparambil, S. Bratus, A. Shubina, and D. Kotz, "On the reliability of wireless fingerprinting using clock skews," in *Proc. ACM WiSec'10*, 2010.

[2] P. Bahl, R. Ch, J. Padhye, L. Ravindranath, M. Singh, A. Wolman, and B. Zill, "Enhancing the security of corporate Wi-Fi networks using DAIR," in *In Proc. ACM MOBISYS*, 2006.

[3] S. Bratus, C. Cornelius, D. Kotz, and D. Peebles, "Active behavioral fingerprinting of wireless devices," in *Proc. ACM WiSec '08*, 2008.

[4] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proc. ACM MobiCom '08*, 2008.

[5] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. Van Randwyk, and D. Sicker, "Passive data link layer 802.11 wireless device driver fingerprinting," in *Proc. 15th USENIX Security Symposium*, 2006.

[6] K. Gao, C. Corbett, and R. Beyah, "A passive approach to wireless device fingerprinting," *IEEE Dependable Systems and Networks*, 2010.

[7] J. Hall, "Enhancing intrusion detection in wireless networks using radio frequency fingerprinting," in *Proc. IASTED CIIT*, 2004.

[8] J. Hall, M. Barbeau, and E. Kranakis, "Detection of transient in radio frequency fingerprinting using signal phase," in *Proc. WOC 2003*, 2003.

[9] *IEEE Standard 1193-1994. IEEE Guide for Measurement of Environmental Sensitivities of Standard Frequency Generators*, IEEE Computer Society, 1995.

[10] *IEEE Standard 802, Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Computer Society, 2007.

[11] S. Jana and S. K. Kasera, "On fast and accurate detection of unauthorized wireless access points using clock skews," in *Proc. ACM MobiCom'08*, 2008.

[12] T. Kohno, A. Broido, and K. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, 2005.

[13] D. L. Mills, "Network time protocol (version 3): Specification, implementation and analysis," Internet Engineering Task Force: RFC 1305, 1992.

[14] ——, *Computer Network Time Synchronization*. CRC Press, 2011.

[15] S. B. Moon, P. Skelly, and D. F. Towsley, "Estimation and removal of clock skew from network delay measurements," in *INFOCOM*, 1999.

[16] S. J. Murdoch, "Hot or not: Revealing hidden services by their clock skew," in *Proc. ACM CCS'06*, 2006.

[17] Network Working Group, "RFC 1323: TCP Extensions for High Performance," May 1992.

[18] C. Rigney, S. Willens, A. Rubens, and W. Simpson, "Remote authentication dial in user service (radius)," Internet Engineering Task Force: RFC 2865, 2000.

[19] D. Shaw and W. Kinsner, "Multifractal modelling of radio transmitter transients for classification," in *Proc. IEEE WESCANEX 97*, May 1997.

[20] B. Sieka, "Active fingerprinting of 802.11 devices by timing analysis," in *Proc. IEEE CCNC'06.*, 2006.

[21] O. Ureten and N. Serinken, "Detection of radio transmitter turn-on transients," *Electronics Letters*, vol. 35, no. 23, pp. 1996 –1997, Nov. 1999.