Experience with Heterogenous Clock-Skew based Device Fingerprinting

Swati Sharma
Dept. of Comp. Sc. & Engg.
Indian Institute of Technology
Delhi, India
sswati@cse.iitd.ernet.in

Alefiya Hussain
Information Sciences Institute
University of Southern
California
Los Angeles, CA, USA
hussain@isi.edu

Huzur Saran
Dept. of Comp. Sc. & Engg.
Indian Institute of Technology
Delhi, India
saran@cse.iitd.ernet.in

ABSTRACT

The goal of this research is to validate clock skew based device fingerprinting introduced in 2005 and explore the feasibility of its usage and/or modification to facilitate unique device identification across heterogenous target devices with improved accuracy and reduced errors. Our network consists of 212 devices that include desktops, laptops and handhelds. We conduct a systematic evaluation of the clock-skew fingerprint stability across 3 primary dimensions namely, change in target host environment, configuration and measurement methodology. We also investigate parameters that affect clock skew of a device. Our results indicate that a minimum of 70 packets are required to achieve a stable skew estimate. We also observe a significant difference between desktop and handheld clock-skew behavior with the factors affecting skew estimates being handheld power state and NTP updates. Thus, for a moderate-size network, clock skew based fingerprints provide a stable and conclusive means of identification for desktops and laptops but show jumps for the handhelds.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network management, Network monitoring*; C.4 [**Performance of Systems**]: Measurement techniques, Performance attributes.

General Terms

Experimentation, Measurement, Reliability, Security

Keywords

Device fingerprinting, clock skew, ICMP, TCP timestamps.

1. INTRODUCTION

Clock skew based fingerprinting technique allows uniquely identifying a physical device on the network. Host identification today can be done at many layers of the network protocol stack. Traditionally, unique host identification has been done by using the MAC addresses or static or dynamic IP addresses. Another strategy for host identification is based on difference in host's response to non-standard packet types or by extracting information from standard packet type responses. Typically, these can be broadly classified as operating system fingerprints [12], browser fingerprints [4], MAC sequence numbers based fingerprints [6], active behavioral fingerprints [1] and fingerprints from other identification parameters from the packet headers of the network protocol stack [2, 3, 7, 8]. We choose to investigate unique device identification based on clock skew fingerprints as the other aforementioned techniques either require special hardware for implementation or there exist trivial countermeasures to spoof their operation [11, 5] and incorporate intolerable errors into it. Clock skew based fingerprint identifies a device based on the negligible shift in system clocks and hence, provides a more robust alternative that is complex to spoof as it depends only on the system hardware. It is harder to spoof rather than traditional or other alternatives. The system time is collectively affected by the kernel standards, target host hardware and target host configuration. Although it is still possible to spoof this clock skew based identification process by altering the system timestamp values, it is a complex procedure and requires a detailed knowledge of how the kernel interprets system time and how this system time is reflected in the default OS generated fields in packet headers.

Clock skew based device identification was introduced by Kohno [8]. This technique utilized a constructive exploitation of the shift in system clock, to identify devices uniquely. Prior to this approach, Moon [9] and Paxson [10] showed that clock skews were only a source of distortions in network measurements and accuracy in network protocols and network measurements necessitated synchronization of clocks for all devices in the network.

Device identification and tracking has numerous applications. For instance, some of the major applications amongst others include computer forensics and attack attribution, monitoring active hosts in the network and intrusion detection for all these active hosts once their normal behavior has been established.

In this paper, we validate the clock skew based device fingerprinting approach that was presented by Kohno et. al. [8]. We plan to achieve this goal by extensive evaluation of the stability of clock skews in order to robustly identify

LASER '12 July 18-19, 2012, Arlington, Virginia USA

a wide range of desktops, laptops, and resource constrained handheld devices in a network. The clock skew based fingerprinting approach is described in brief in the next section. With this goal in mind, our first step in this direction was to determine if the device fingerprint was stable and portable across environmental conditions, device configurations and measurement methodologies. Once this was done, it was trivial to determine the factors which affected the target device skew estimates by eliminating the ones under which the skew estimate was stable. We experimented with a large set of devices including shared and private machines consisting of desktops, laptops, netbooks, macbooks, iMacs, virtual machines, smartphones and tablets. In this text, we do not discuss the skew behavior of virtual machines. We tracked these devices under a heterogenous set of contrasting conditions for a period of 9 months from June 2010 to Feb 2011 in order to study their processor clock skew. All servers and machines from four different undergraduate and graduate student computing laboratories in the computer science department at the Indian Institute of Technology in Delhi, India were a subset of this target device set. Several of these machines were identified and then tracked when they first entered the network.

The contribution of this work is a systematic evaluation of the stability of clock skew based device fingerprint and the factors influencing it in a heterogenous environment. During the nine month period, we fingerprinted and tracked 162 devices and 48 virtual machines by performing a multi-dimensional comparison of factors influencing clock skews. These dimensions were (a) measurement techniques, (b) target host measurement environments, and (c) target host configurations. The deviations from Kohno's measurement techniques included modifications in ICMP timestamp collection process, variable length measurement intervals between timestamp collections, temperature variations at target devices ranging from 104 to 140 degree fahrenheit, power source variations from battery-backed to AC power operated, variations in operating systems on same target device, variations in system state, device configurations, network connectivity and vantage points while calculating the skew estimate of a target device. The rest of the paper is organized as follows. Section 2 briefly describes the clock skew based device identification approach and discusses how to interpret the observations. Section 3 gives the detailed list of scenarios under which the clock skew was found to be stable and a description of their interpretations for each subcategory. Finally, section 4 concludes our observations on clock skew measurements and summarizes our findings.

2. METHODOLOGY

In this section, we first discuss some details about the skew fingerprint technique and how this work differs from the original version. This is followed by the definitions given by Paxson [10] and Kohno [8] to introduce the terminology used in this paper for describing a clock's behavior. Subsequently, we give a brief summary of skew extraction process accompanied by a brief account of our experimentation setup. Finally, we illustrate how to interpret the observations and extract skew values from them to identify devices.

2.1 Previous work on clock skew fingerprints

The concept of device identification introduced by Kohno et. al., was based on *exploitation of small, microscopic deviations in the device hardware*, that is clock skews. They remotely identified fingerprintees after analysis of 12 hour and 24 hour trace periods. They also applied the technique to devices in a trace collected at one of the US's Tier 1 OC-48 links. The identification was based on accurate extraction of system timestamps from the TCP and ICMP packets for clock skew estimation. They also demonstrated that clock skew was independent of the system hardware by remotely fingerprinting 69 homogenously configured machines uniquely at UCSD's undergraduate computing laboratory. These fingerprintee machines were Micron PC's with 448 MHz Pentium II processors running Microsoft Windows XP Professional Service Pack 1. The major contribution of their work was introduction of a new approach to enable remote unique identification of a device on a network, outlining applications that could benefit from this approach in the process. In this paper, we present a validation of this approach and a detailed analysis of factors affecting the clock skew based identification process for hosts in a network. We diversify our set of target devices to include heterogenous device kinds, that is desktops - clients and servers, virtual machines, laptops, netbooks and resource constrained mobile handheld devices, as illustrated in Figure 1. We further explore the feasibility of unique device identification in such a network.

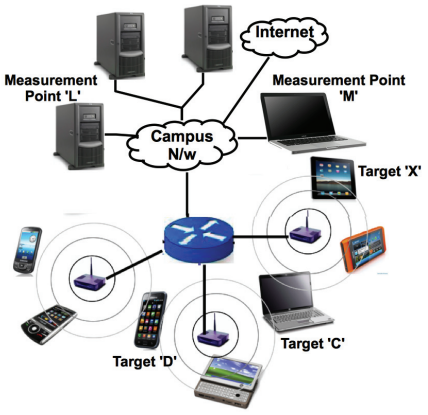

Figure 1: Network layout of the experimental setup illustrating multiple measurement points and a diverse set of target devices.

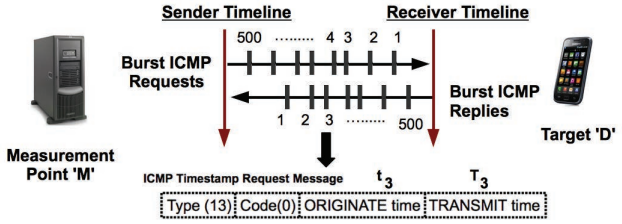

Figure 2: Illustration of batch ICMP timestamp collection mode in brief.

2.2 Terminology Used

A clock's *resolution* is defined as the smallest unit of time by which the clock time can increase. *Accuracy* of a clock is how well the frequency, and hence, time can be compared with the true time. The time represented by the clock is

S.No.	Descriptor	Values
1	Numbers	152 nodes, 48 virtual machines, 10 handhelds, 19 operating systems.
2	Devices	Nodes: 91 Desktops, 20 Servers, 15 iMacs, 17 Laptops, 5 Macbooks, 4 Acer Netbooks. Handhelds: 2 iPads, 4 Samsung Galaxy GT-I9000, 2 Nokia N8's, 2 Dell XCD28's.
3	Operating Systems	Fedora core 10, 11, 12, freeBSD 5.2.1, Ubuntu version 8.04, 9.10, Windows XP (SP 1, 2, 3), Windows Vista, Windows 7, Macintosh OS X version 10.5.8, 10.6.2 and 10.6.4, Android OS version 2.1, 2.2, Android OS x86, Symbian $\hat{3}$, iOS version 4.0.
4	Measurement Method	Active ICMP based technique, Passive TCP based technique.
5	Measurement Intervals	0.1 sec, 1 sec, 1 min, 2 mins, 5 mins, 10 mins, 20 mins, 30 mins, 40 mins, 60 mins.
6	Measurement Modes	Continuous ICMP, Batch ICMP, TCP.

Table 1: Experimental Configuration for clock skew measurement conducted at IIT Delhi from June 2010 to February 2011.

called the *reported time*. *Stability* of a clock is how well can it maintain a constant frequency at which the oscillator feeding the motherboard circuits oscillates. The *offset* of a clock is the difference between reported time and true time at any particular moment. Thus, offset between any two given clocks can be interpreted as the difference in the reported times of the respective clocks. A *clock's skew* is defined as the first-order derivative of its time offset with respect to the true time or the reported time of another clock. It can be interpreted as the time difference between two machines for every second of true time that elapses, and is measured in ppm or parts per million. Thus, a skew value of 2 ppm between two machines can be interpreted as a difference of 2 microseconds in the reported times of the corresponding clocks, for every second that elapses. This amounts to a difference of a few seconds in one day.

System clocks exhibit a negligible shift with time and these shifts are characteristic of the frequency at which the oscillator feeding the motherboard circuit oscillates. Clock skews are, thus, an *inherent* property of the device that can be used for fingerprinting both wired and wireless devices irrespective of the device's hardware/software configurations and the network conditions. We now discuss the skew estimation procedure.

2.3 Clock Skew Extraction

The precision of the skew calculation lies in the accurate extraction of system time generated by the oscillator frequency. Kohno et. al. [8] showed that TCP timestamps from the TCP header and ICMP timestamps from the ICMP timestamp response header provide the adequate granularity for this purpose. TCP timestamps were captured from TCP communication taking place between the main server and clients. Thus, it was a passive measurement technique where the client initiated connection and the server just responded to it. For ICMP timestamps, the server initiated communication in the form of ICMP Timestamp Request messages and extracted the target machine timestamps from ICMP Timestamp Response messages. Since, the server initiated the communication, this was an active fingerprinting technique.

Equations 1 through 5 below explain the skew estimation procedure. Arrival time is the time at which a packet is received at the network measurement point, M and the target time is the time at which a packet left the target or destination, D. For ICMP timestamps, the arrival time is extracted from the *ORIGINATE* time field of the ICMP Timestamp Request header, while the target time is extracted from the *TRANSMIT* time field of the ICMP Timestamp Response header as illustrated in Figure 2. For the TCP timestamps, the arrival time is timestamped by M, when the packet is captured by the NIC and the target time is extracted from the TCP timestamp field in the TCP header.

Let t_1 denote the timestamp from the first packet and t_i denote the timestamp from the i^{th} packet at the measurement point. Similarly, let T_1 denote the timestamp from the first packet and T_i denote the timestamp from the i^{th} packet at the target device. Let m_i denote the arrival time offset, calculated in equation 1. Likewise, let d_i denote the target time offset upto the i^{th} packet, calculated in equation 2. We normalized the target timestamp offset, d_i with the target's Interrupt Timer Frequency, *Hz*. This was necessary in order to make fair comparison of skew values across the diverse target device set and ensure the statistical soundness of the method. The observed clock offset between the measurement point and the target clocks at the measurement point was then calculated as the difference between these two time offsets, as in equation 4. The observed time offset between the two clocks is given by o_i and the clock skew, s_i is computed as the first-order derivative of this time offset value in offset set OT.

$$m_i = t_i - t_1, \tag{1}$$

$$d_i = T_i - T_1 \tag{2}$$

$$n_i = \frac{d_i}{Hz} \tag{3}$$

$$o_i = n_i - m_i \tag{4}$$

$$OT = \{(m_i, o_i) | i = 1, ..., |D|\}. \tag{5}$$

ICMP timestamps could not be recovered from the target machines running Windows. This is because all versions of the Windows operating system embed time from a virtual clock rather than the system time, into the timestamp fields. The virtual clock may or may not be related to the system time in different variations but maintains the flawless functioning of the RTT and PAWS mechanisms by preserving a

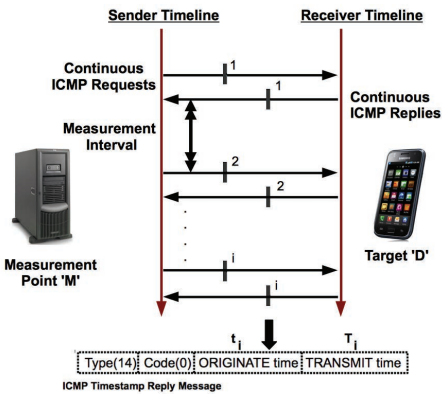

Figure 3: Illustration of continuous ICMP timestamp collection mode in brief.

notion of sequence between these random timestamp values. Also, ICMP timestamps are disabled by default in all Windows and Macintosh machines, so they have to be enabled first, before any of these measurements can be performed.

In our measurement methodology, we simply repeated the TCP timestamp extraction procedure outlined by Kohno but modified the ICMP timestamp collection procedure to achieve two variations: batch mode and continuous mode. In the batch mode, a burst of packets is sent to the target after regular intervals of time. The measurement point waited for the responses once the burst had been sent, in a blocking fashion, that is sitting idle and waiting for all responses to be received. We sent a burst of 500 packets in 50 seconds at a time or 10 packets in 1 second. Once responses to this packet burst were received, timestamp values were extracted from the packet corresponding to the minimum RTT (Round Trip Time), maximum RTT and a stable RTT packet in order to inspect the network conditions and delays. From this, the stable RTT value was used for skew calculation. On the other hand, in continuous mode, the measurement point sent an ICMP timestamp request packet to the target node at regular intervals of time. This was done in an unblocking fashion, that is the measurement point did not wait for a response after a request had been sent. Figures 2 and 3 illustrate this difference in the batch and the continuous modes of ICMP skew extraction method. In both these modes, we sent packets after regular intervals of time called *measurement intervals*. These regular measurement intervals corresponded to 0.1 sec, 1 sec, 1 min, 2 mins, 5 mins, 10 mins, 20 mins, 30 mins, 40 mins and 1 hour.

Since, TCP timestamp collection was a passive process, as soon as the measurement point encountered 10 packets belonging to the same host but with increasing sequence numbers, it started capturing them. It did not wait for the flow to finish in order to capture it completely. Rather, as soon as the required number of packets were accumulated, the capture terminated and the timestamp extraction procedure for skew estimation commenced. Whether the measurement methodology for timestamp collection was TCP or batch ICMP or continuous ICMP, 70 packets were used for one skew estimate. The discussion on the significance of this number follows in the next section.

We also demonstrated that skew estimates for target hosts were independent of the measurement point. The measurement point basically captured packets to extract timestamps or reported target clock times and then calculated the clock skew for all hosts in the network. Clock skew estimates for target devices are independent of the measurement point's clock shift as skew is a relative quantity, relying on two machines. Thus, making one machine universal as the measurement point eliminates the effect of measurement point's clock shift on the skews of target devices. But, if we consider two measurement points *L and M* illustrated in Figure 1, clock skew for a target machine *X* was different with respect to L than with respect to M, and this difference was exactly the same as the clock skew of L with respect to M. We use two measurement points to confirm this notion, details of which are presented in section 3.2.

Variation	Range, Error Thresholds
Measurement Mode	Batch : 0.1 ppm Continuous : 0.2 ppm
Measurement Interval	0.1 s to 60 mins : 0.2 ppm
Single Capture Duration	Hours to days : 0.2 ppm
Total Capture Duration	June'10 to Feb'11 : 0.3 ppm

Table 2: Clock Skew stability across different investigation parameters of system timestamp collection methodology.

2.4 Experimental Setup

Figure 1 illustrates the general topology for experimental setup of our experiments. Briefly, it consists of target devices and measurement points in the network with respect to which clock skews for all the target devices were calculated. These devices were connected in a client and server fashion with heterogenous composition of target devices, as seen in Figure 1 and summarized in Table 1. The measurement point in a network can be an access point for a wireless network or a network controller monitoring the whole network, which has visibility of all packets travelling over the network. Measurement point *'L'* is a server machine with a dual CPU Xeon processor, 24 GB RAM and running 32-bit kernel version 2.6.26 of Fedora Core 8. Measurement point *'M'* is a Macbook with 2.13 GHz Intel Core 2 Duo processor, 2 GB DDR2 RAM and running Mac OS X version 10.5.8. NTP daemon at the measurement point and target devices was shut down several hours before the start of measurement process. The affect of the NTP synchronization on target clock skews was monitored by switching it on and off on the target hosts at different time instants.

Our set of target machines included 152 desktops, 48 virtual machines and 10 handheld devices. This device set was a mix of personal and shared campus network hosts and consisted of servers and machines in four undergraduate and graduate computing laboratories in the Computer Science and Engineering department at the Indian Institute of Technology, Delhi, India. Targets were 91 Desktops, 20 Servers, 15 iMacs, 17 Laptops, 5 Macbooks, 4 Acer Netbooks, 2 iPad's, 4 Samsung Galaxy GT-I9000's, 2 Nokia N8's and 2 Dell XCD28's. The operating systems running on the target devices were Fedora core 10, 11, 12, freeBSD 5.2.1, Ubuntu version 8.04, 9.10, Windows XP (SP 1, 2, 3), Windows Vista, Windows 7, Macintosh OS X version 10.5.8, 10.6.2, 10.6.4, Android OS version 2.1, 2.2 for Sam-

Variation	Range, Error Thresholds
Temperature Variations	Network load, 95% resource utilization : 0.1 ppm
Network Connectivity	wired/wireless/edge : 0.1 ppm
Measurement Point	2 different machines : 0.2 ppm

Table 3: Clock Skew stability across different investigation parameters of target host environments.

sung Galaxy GT-I9000 Android phones, Android OS x86 for Netbooks, Symbian $\hat{3}$ for Nokia N8 phones and iOS 4.0 for iPad. The duration of these measurements was from June 2010 to February 2011 and during this period the measurements were repeated every month. These details are summarized in Table 1. All the mobile devices, that is laptops, netbooks, smartphones and tablets were completely stationary during the measurement period in the experimentation process.

2.5 Interpretation of Observations

Consistent measurements were reported for the skew values over the Internet by Kohno. So, we expected to observe a stable skew value within an error range for every target device. Figure 4 shows the skew behavior of a target device for a duration of approximately six hours. The x-axis in the figure depicts the measurement point timestamps or arrival time m_i as calculated in equation 1, while the y-axis depicts the observed offset between the target device and the measurement point in seconds, as calculated in equation 4. Note that the x-axis does not start from 0 as the measurement point was left idle for the first 38 hours with NTP time updates switched off so that the measurement point could achieve a notable difference in its reported time with respect to the true time. The first-order derivative of the time offset is the clock skew, and it is conveyed by the slope of the line. Thus, a straight line shows a consistent slope and hence, a stable skew value. We calculated skews for a set of 16 homogenous machines which had exactly same hardware and software configuration. But, their skews were unique which was demonstrated by distinct lines corresponding to each target device, thus having a different slope and hence a unique clock skew value.

Let us now understand the concept of error threshold. If the clock skew value for a particular target *'X'* is 18.96 ppm then it signifies that for every one second elapse of true time, there is a difference of 18.96 microseconds between the clocks of measurement point and target X. Clock skew of 18.96 ppm with an error threshold of say 0.3 ppm can be interpreted as the skew value falling in the range of 18.96 ppm plus/minus 0.15 ppm for all future measurements for the same target. Once a new host was identified in the network, its skew value was written into a database for further comparisons facilitating host identification in a real time operation mode. Thus, machines were identified based on their clock skew values. The stability of clock skews under a wide range of contrasting conditions is discussed in the next section.

3. CLOCK SKEW STABILITY

With the proliferation of mobile devices in modern networks, every network consists of a heterogenous mix of hosts consisting of desktops, laptops, resource constrained net-

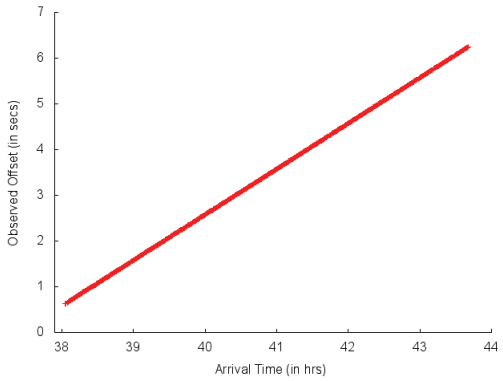

Figure 4: Interpretation of skew behavior from measurement point and target host timestamps collected from ICMP timestamp response packet headers for device ID 4.

Variation	Range, Error Thresholds
Power State	Battery operated or not : 0.2 ppm
Operating Systems	0.1 ppm for smartphones, 0.3 ppm for desktops
Time Synchronization	0.1 ppm for handhelds
System State	Reboots, sleep periods : 0.1 ppm

Table 4: Clock Skew stability across different investigation parameters of target host configuration.

books, and critically resource constrained handhelds like smartphones and tablets. All these devices should be uniquely identifiable irrespective of their configurations. In this section, we share data and observations from a subset of our target device set consisting of 42 machines and 10 handheld devices, but the observations were universal to the entire target device set. The detailed breakup of these 52 machines is shown in Table 5. *Homogenous Configuration* refers to same hardware and software configuration for all the target devices in that device ID range. Similarly, *Heterogenous Configuration* for a device ID range indicates differences in the hardware and/or software configuration of the target devices in that device ID range. All nodes had multiple operating systems installed so that the effect of different software configuration and same hardware configuration on clock skew value could be monitored. There were machines that were shared amongst many users in the lab, so they ran other processes apart from entertaining the TCP and ICMP request-response communications for skew calculations. Similarly, all the handheld devices except iPads were shared experimental devices belonging to the laboratory. In this section, we perform a multi-dimensional comparison of factors to observe their effects on the clock skew behavior, with these dimensions being (a) measurement methodologies, (b) measurement environments, and (c) target host configurations.

3.1 Across Measurement Methodologies

In this subsection, we investigated three different variations in the timestamp collection methodologies, but thereafter, the skew calculation process remains the same. These variations were : (1) impact of different ICMP measurement

Device ID Range	Device Quantity	Device Type	Configuration Type
1 to 4	4	Samsung Galaxy	Homogenous
5 to 6	2	Nokia N8s	Homogenous
7 to 8	2	Dell XCD28s	Homogenous
9 to 10	2	iPads	Heterogenous
11 to 21	11	Desktops	Heterogenous
22 to 25	4	Laptops	Heterogenous
26 to 41	16	Desktops	Homogenous
42 to 47	6	iMacs	Homogenous
48 to 50	3	Macbooks	Heterogenous
51 to 52	2	Netbooks	Homogenous

Table 5: Device identification details for the target device subset illustrated in figure 6.

modes, (2) impact of varying measurement intervals between consecutive packet captures, and (3) impact of single and total capture duration. Our results indicated that the maximum error threshold observed amongst all the subcategories was in the total capture duration spanning 9 months, and was 0.3 ppm as shown in Table 2.

1. **Across ICMP Measurement Modes**: We used two modes of ICMP measurements : batch and continuous, comparison between them is illustrated in Figures 2 and 3. The difference in these two modes lies in how much do they allow latencies in the underlying physical network to affect their respective skew calculations. For instance, let us consider a regular measurement interval of 1 minute after which the measurement point sent a single packet in continuous mode and packet bursts in batch mode, to the target device for timestamp extraction. Under the worst circumstances, further assume that the packets were experiencing considerable latency due to congestion in the network. The effect of this latency can be observed in the varying RTT values in the ICMP timestamp response packets received from the targets. Granularity of ICMP timestamps is in milliseconds, so latency variation of even a few milliseconds might result in distortion in the range of skew values. In the continuous mode, latency in the network might affect the consistency of the timestamp values, that is the calculated skew might or might not lie in the same skew range as the previous calculations because of the delay experienced by the packet in reaching the measurement point. But, in the batch mode, the measurement point sent a burst of packets to the target and waited for responses. It then selected the packet corresponding to minimum RTT, maximum RTT and a stable RTT to get a sample of the latency in network during that packet burst, between M and D. The timestamp values from this packet with a stable RTT was then used for skew calculations which resulted in a more accurate range of skew values. This can be verified by comparing the error thresholds for batch and continuous ICMP modes for the same set of target devices on the network. While the maximum error threshold for the continuous mode is 0.2 ppm, the maximum error threshold for the batch mode is 0.1 ppm, resulting from more precise skew values, summarized in Table 2.

Measurement Interval	Standard Deviation Device ID 2	Standard Deviation Device ID 9
60 mins	39.3	46.27
30 mins	32.03	35.92
5 mins	27.7	30.29
1 sec	24.8	26.17

Table 6: Illustration of skew value scatter for different measurement intervals for device ID 2 and ID 9.

2. **Across Measurement Interval of Capture**: Measurement interval is the intervening period between sending two consecutive packets in the continuous ICMP mode or two consecutive bursts in the burst ICMP mode at the measurement point. The measurement point sat idle during this interval and the value of this interval varies from 0.1 seconds to 60 minutes as shown in Table 1. We observed a maximum error threshold of 0.2 ppm in the skew values for any particular device when regular measurement intervals of 0.1 sec, 1 sec, 1 min, 2 mins, 5 mins, 10 mins, 20 mins, 30 mins, 40 mins and 60 mins were inserted in between packet captures as explained above. We also observed that with an increasing value of the measurement interval between the consecutive packet captures, the scatter of skew values in the same range changed. This can be observed from the standard deviation of the skew values for each of these measurement intervals. Observations from a sample data set for an iPad and a Galaxy smartphone are shown in Table 6 where the device ID's are from Table 5. As the length of the interval increases from 1 sec to 60 mins, the standard deviation of the skew values also increases, thus illustrating a wider scatter of skew value points for a larger measurement interval.

3. **Across Capture Duration for Single Skew Estimate**: The packet capture duration for a single ICMP skew estimate was variable and was dependent on the measurement interval between consecutive packets sent from the measurement point, M. For instance, if a set of 100 timestamps is used for one skew estimate of the target device, the capture duration can be 100 minutes corresponding to 1 minute measurement intervals or 100 hours corresponding to 1 hour measurement intervals. On the other hand, for TCP measurements, the same set of 100 timestamps was yielded in only a few minutes of data capture depending on the network load.

   Now, a single skew estimate can be computed by a variable number of system timestamps. This calculation can be done from as few as 10 timestamps and from as large as 100 timestamps. The only difference in the two scenarios is the level of accuracy achieved and the precision of the calculated skew value. This is shown in figure 5 by illustrating data from a Samsung Galaxy GT-9000 smartphone. In both these subfigures, x-axis denotes the number of packets used for skew calculations and the y-axis denotes the skew estimate or the error threshold, respectively. We observed that in order to achieve a stable, accurate clock skew value and

a low error threshold, more number of timestamped packets must be taken into consideration. This requires a longer data capture duration and hence, more overheads will be incurred to identify a device in a real time application. On the other hand, if lesser number of packets are used for the clock skew estimation comparatively, then the value of error threshold will be high. This affects the skew accuracy at the measurement point adversely, and the number of false positives increase as more skew values lie in the same range due to larger error thresholds.

In figure 5, when only 10 packets were used for the skew estimate, the error threshold was 0.08 ppm with fluctuating skew values. But as soon as the number of timestamps utilized for the skew estimate were increased to 40, the error threshold reduced to 0.04 ppm. And when the number packets was 70 or more, the error threshold was reduced to its minimum value of 0.001 ppm. We thus conclude that although skew calculation can be done from as few as 10 timestamp values, a minimum number of 70 system timestamps is required for the skew estimate to become stable resulting in minimum error threshold.

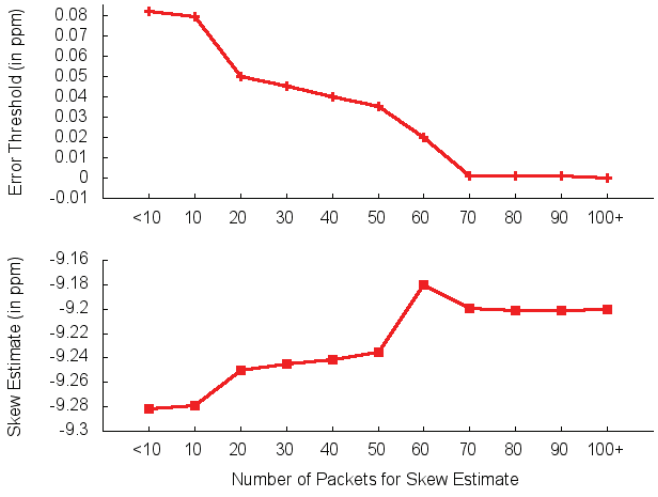

Figure 5: Error Threshold and Skew Estimate variation with respect to number of packets utilized for skew computation.

4. **Across Total Data Capture Duration**: The measurements were repeated every month from June'10 to Feb'11 to investigate the stability of the clock skew with respect to time. The data collection for all devices was distributed uniformly over the 30 day period so as to put enough packet load on the network and at the same time not to cause excessive computation delays on the measurement point due to extra network load and CPU load. The maximum error threshold observed in the skew calculations for the described device subset over the 9 month duration was 0.3 ppm as summarized in Table 2.

3.2 Across Target Device Environments

In this subsection, we discuss the different environments to which the target device is subjected to for the duration of timestamp collection process. We investigated three distinct variations in the target device environment to monitor their effect on clock skew behavior : (1) impact of temperature variations in target device caused due to system overloads, (2) impact of network connectivities, and (3) target device clock skew dependence on the measurement point. Our results indicated that the maximum error threshold observed amongst all these subcategories was when skews were calculated with respect to different measurement points, and was 0.2 ppm as summarized in Table 3.

1. **Across Temperature Variation**: The clock skew based fingerprinting technique derives its unique diagnostic behavior from the oscillator that powers the clock. But, oscillator operation is affected by the operating temperature of the device. Thus, the clock frequency and hence the precision of the fingerprinting technique in an air conditioned indoor environment should not be the same as that in a hot environment. Our analogy of this temperature variation for a hot environment in the wired machines was created by the processor operating under full load with more than 95% CPU utilization under heavy network loads, thereby increasing the average operating temperature for the oscillator in the motherboard that seeds the system time. Under this subcategory, we did repeated measurements under stable temperature, stable load environment and temperature varying, maximum load environment. This load behavior of the devices was confirmed by continuous monitoring of the device by the resource manager. The conditions of heavy network load were created by a mixup of large file downloads, ICMP communication, bursts of web traffic in the devices and additional online activity of various apps in the handheld devices. The conditions of heavy CPU utilization were created by running heavy computation based scripts in infinite loops along with multiple processes, in order to achieve a CPU utilization of 95% or more and a heavy memory utilization. This brought about clearly observable changes in the temperature of the device and hence in the working temperature of the oscillator. We observed a maximum error threshold of 0.1 ppm due to variation in skew values caused by temperature changes in all target devices. We can summarize the temperature variations by stating that the skew value was stable in the device operation range. We plan to explore more in this direction by observing the effect of more drastic temperature changes on the target device clock skews.

2. **Across Network Connectivities**: In order to be able to uniquely identify machines on a network, the skew calculation should be resistant to link fluctuations/delays in the network and type of network connectivity. This concern was addressed partially in the methodology for batch mode in the first measurement scenario discussed in section 3.1. We repeated our experiments to measure clock skew when the same device was connected to the measurement point with wired or wireless connectivity for the desktops, laptops and netbooks and with wifi or edge connectivity for the smartphones. We observed a negligible maximum er-

Device ID	Skew (at L)	Error (at. L)	Skew (at M)	Error (at M)
2	-19.75	0.1	64.63	0.1
4	-16.08	0.1	68.3	0.1
8	-28.49	0.2	55.97	0.2
9	-4.82	0.2	79.55	0.2

Table 7: Illustration of dependence of skew behavior on measurement points.

Device ID	AC Skew	AC Error	Battery Skew	Battery Error
2	-19.75	0.1	-18.7	0.1
4	-16.08	0.1	-15.23	0.1
8	-28.49	0.2	-27.57	0.2
9	-4.82	0.2	-3.90	0.2

Table 8: Illustration of dependence of skew behavior on the power state of handhelds.

Device ID	Android OS 2.1	Android OS 2.2
2	-19.75	-19.76
4	-16.08	-16.14

Table 9: Illustration of dependence of skew behavior on target device operating systems.

ror threshold of 0.1 ppm because of the above mentioned changes in the network connectivity.

3. **Across Reference Measurement Points**: Measurements for all the investigation parameters were repeated for two measurement points on the network, L and M, illustrated in Figure 1. These machines had a global network view and access to all the packets being transported in the network. Of these two measurement points, one was a server machine with ample memory and computation resources, while the other was a normal laptop running processes solely for the purpose of skew calculations. The skew values for all target devices were stable across both the measurement points, but their mathematical value was different for both as expected. The rationale behind this being that skew is the first order derivative of the time offset between two machines. So, when one of the machines that is measurement point changes, the skew value is bound to change. And this change is exactly equal to the skew between the two measurement points L and M. This is illustrated in Table 7 where the device ID's are from Table 5 and the skew estimates and error thresholds are reported in ppm. The skew between measurement points 'L' and 'M' was found to be -84.4 ppm. The table clearly illustrates that the difference in the skew values for any particular target device is exactly the skew value between the two measurement points. The maximum error threshold in these skew calculations was found out to be 0.2 ppm.

3.3 Across Target Device Configurations

In this subsection, we discuss the variations in the hardware and software configurations at the target device. We investigated the effect of four distinct configuration parameters on the clock skew behavior: (1) variations in target device power state that is battery-run or AC power backed, (2) variations in the operating systems of the target hosts, (3) impact of presence of time synchronization, and (4) variations in the system state, that is sleep periods and system reboots. Our results indicated that the maximum error threshold observed amongst all these subcategories was with the variations in operating systems. This threshold was 0.3 ppm for the desktops, laptops and netbooks and 0.1 ppm for the resource constrained handheld devices, as summarized in Table 4.

1. **Across Power State**: This investigation parameter leads to two contrasting situations, battery backed and AC power backed operation. Smartphones and tablets are resource constrained devices in terms of CPU, memory, etc and contain optimizations in their operating systems for conserving these resources for optimal usage. It was interesting to observe changes in the clock skew values when the device under consideration was operating on a battery as opposed to being connected to an AC power supply. We also observed a variation in the sleep schedules and activity monitoring on these resource constrained devices when they were operating on AC power as opposed to when they were not. Some of the data for these repeated measurements is shown in Table 8 where the device ID's are from Table 5 and the skew estimates and error thresholds are reported in ppm. We observed a maximum error threshold of 0.2 ppm in the clock skew values for each of these measurements for all devices. Further, we observed a difference of about 1 ppm in the clock skew values when the device was operating under these contrasting circumstances. We are currently investigating the possible causes for this observation.

2. **Across Operating Systems and/or different kernel versions**: To illustrate the effect of this investigation parameter on desktops and other laptops, we installed multiple operating systems and different kernel versions of the same operating system on the devices. For the smartphone counterparts, we upgraded the operating systems to change the kernel version of the Android OS. Since, skew is characteristic of the oscillator, we expected to see stable clock skews across these environments for the same device. We present the observed data for two Samsung Galaxy smartphones in Table 9 where the device ID's are from Table 5. All the skew estimates are in ppm and the error thresholds for all these devices were 0.1 ppm. These devices had exactly same hardware and software configurations and possess unmistakably different skew values. Also, variations in the operating system versions did not bring about much change in the skew values or the error threshold. To recapitulate, error threshold is the range in which the data points are scattered above and below the calculated skew value for a particular target device. We observed an error threshold of 0.1 ppm for the handhelds and 0.3 ppm for the desktops and laptops under this scenario.

3. **Across Time Synchronization on the Mobile De-**

vice: The presence of time synchronization daemon in a machine synchronizes the system clock to another time server in the network at regular intervals of time. Ideally, this behavior should bring distortions in the skew estimates. In this experiment category, skew measurement for all the target devices was carried out in the presence of and absence of time synchronization. The observed skew behavior for desktops and laptops was exactly as expected. ICMP timestamp values were garbled due to frequent NTP updates at the measurement point and the target device, thus, disturbing the skew calculations. TCP timestamps on the other hand were not affected by the NTP updates. But, we observed that the presence of time synchronization on the handheld devices did not affect the precision of skew values or their error thresholds. We found the maximum error threshold in this case to be 0.1 ppm.

4. **Across System Reboots and Sleep Periods**: Ideally, the system clock must be consistent for stable clock skews and must not be affected by sleep periods or system reboots. We found that the mechanism of dealing with sleep periods is very different in desktops and laptops as compared to the resource constrained handheld devices and wrote applications for Android OS in Samsung Galaxy, Dell XCD and Symbian OS in Nokia N8 smartphones. These applications made sure that packet capture rates for skew calculations were not compromised and at the same time, limited resources possessed by these devices were also not wasted. In this experiment category, we repeated the clock skew measurements such that system timestamps were collected before and after target device sleep periods and reboots. The apps that we built for smartphones made sure that these sleep periods were of 2 types : (1) uniform and regular, and (2) increasing from a small value like 5 mins to a large one like 5 hours. We found that the maximum error threshold for the skew calculations was 0.1 ppm.

4. DEVICE TRACKING SENSITIVITY

Our observations from the variations in the timestamp collection methodology, target device environment and configurations showed us that the clock skew values were stable within a small error threshold. But, the error threshold varies for each investigation parameter as summarized in Tables 2 through 4. Fluctuating error thresholds impact the ability of the measurement point to accurately identify target devices in the network. Let us establish this statement with our target device subset of 52 machines. Figure 6 illustrates the scatter plot for the skew values of all the machines in this subset when 70 or more packets have been used for the ICMP skew estimation process. The x-axis denotes the device identifications and the y-axis denotes the ICMP skew estimates. Table 5 provides a detailed description of these machines based on device ID's from x-axis in the figure.

Larger is this value of error threshold, more is the probability of multiple target devices from a heterogenous testbed falling in the same skew value range. This leads to collisions or false positives in device identification in that particular skew range. For instance, from Figure 6, we observe that the skew values of device 43 and 44 are very close. If the error threshold is large enough to encompass both these values,

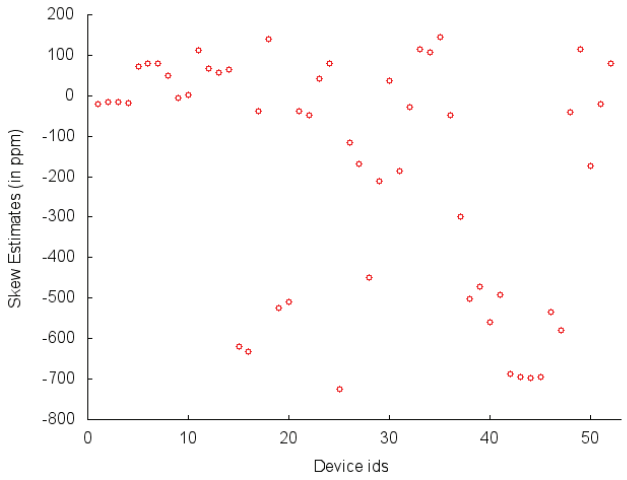

Figure 6: Scatter plot for illustration of skew estimates with respect to the device identifications.

then the aforementioned collision occurs in identification of these devices. But, with a smaller error threshold value, these 2 devices will still belong to different skew ranges, and hence be uniquely identifiable.

While dealing with variations in the timestamp collection methodology, we observed that the minimum error threshold was associated with the batch ICMP collection mode with a value of 0.1 ppm and the maximum error threshold was associated with skew calculations across the total data capture duration with a value of 0.3 ppm. This can be interpreted as follows. Sensitivity in the clock skew based device identification with respect to batch ICMP mode is the maximum, with least false positive rate. The number of false positives in the technique increased as we changed our methodology to continuous ICMP mode, with an error threshold of 0.2 ppm and it increased further with an increment in the capture duration, accompanied by an error threshold of 0.3 ppm. Similarly, for variations in the target device environment, the maximum sensitivity and the least number of false positives corresponded to the temperature variations and variations in the network connectivity with an error threshold of 0.1 ppm. But, with a change in the measurement points, the false positive rate increased with an increase in error threshold value to 0.2 ppm.

The clock skew values for our target device set were uniformly random. Figure 6 just depicts a subset of target devices and is for the understanding of the readers rather than for illustration of the uniform random behavior of the clock skew value for the complete target device set. Let us now quantify these sensitivity values in terms of number of devices. Temperature variations in the target device processor caused an error threshold of 0.1 ppm while operating system variations caused an error threshold of 0.3 ppm in the desktops. Here ppm stands for parts per million, so 0.1 ppm corresponds to 1 part in 10 million parts. Also, let us assume that each skew range is a unique bin which should ideally contain just one device for unique identification. This means that if the error threshold is 0.1 ppm, then for more than one device to fall in the same bin, the total number of devices in the network must be atleast 10 million. On the other hand, if the error threshold is 0.3 ppm, then for

more than one device to fall in the same bin, the minimum number of devices in the network reduces to 3 million. Having said this, in our network, if the error threshold was 0.1 ppm or 0.2 ppm then we were able to accurately identify all 162 machines. But when the error threshold increased to 0.3 ppm, we could only identify 159 devices out of the original 162, for reasons just explained. The process followed was capturing packets, dividing them into IP based bins, extracting timestamps and then evaluating the clock skew from timestamps in each bin. This resulted in very few collisions in a medium size network such as ours consisting of 162 distinct hosts and 2 measurement points. But if the size of the network increases, there may exist an intersection in the clock skew ranges of the target devices owing to the uniform random distribution of skew values, thus resulting in possibly ambiguous device resolution.

We therefore conclude that clock skew based device identification works well for a moderate size network based on our empirical validation of the technique. We found the target device skew to be stable across variations in measurement methodologies, target host configurations and target host environments for desktops, laptops and netbooks. The only factors affecting the skew values were duration of capture and power state and operating system variations at the target host.

Future Work and Limitations: We are currently exploring the factors affecting clock skew in more detail along with the effect of drastic ambient temperature changes on the target device clock skew and the reason for the 1 ppm skew jump with change in the handheld power source. There do exist countermeasures that an adversary can take to escape unique identification, by tampering with the timestamp values inserted in ICMP and TCP packets or by playing with the factors mentioned above that affect the clock skew estimate. The only possibility to thwart such a countermeasure is to club the clock skew based fingerprint with another parameter and combine their results to provide unique identification. One of the major limitations of this work is the absence of hypothesis testing and statistical analysis for all the investigation parameters. We are currently looking into this.

5. CONCLUSION

In this paper, we presented an extensive systematic evaluation of the stability of clock skew based device fingerprinting technique in a moderate size and heterogenous device network. Specifically, we explored the feasibility of device identification on a network comprised of 152 desktops and laptops, 48 virtual machines, and 10 handheld devices. We demonstrated that to achieve an error-free and stable clock skew estimate at least 70 timestamped packets must be captured from a target device. The introduction of batch ICMP mode helped improve the accuracy of the skew estimation algorithm and reduced the maximum observed error threshold to 0.3 ppm. We evaluated the clock-skew across measurement methodologies, target device environments and target device configurations. Our findings indicate that the skew estimate is affected by power state of handhelds, regular NTP updates in desktops, and the capture duration. As the scale of the network increases and the number of skew samples reduce, other techniques may need to be combined with those evaluated in the paper to conclusively distinguish between devices on the network.

6. REFERENCES

[1] S. Bratus, C. Cornelius, D. Kotz, and D. Peebles. Active behavioral fingerprinting of wireless devices. In *Proceedings of the First ACM Conference on Wireless Network Security*, WiSec '08, pages 56–61, New York, NY, USA, 2008.

[2] V. Brik, S. Banerjee, M. Gruteser, and S. Oh. Wireless device identification with radiometric signatures. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, MobiCom '08, pages 116–127, New York, NY, USA, 2008.

[3] L. C. C. Desmond, C. C. Yuan, T. C. Pheng, and R. S. Lee. Identifying unique devices through wireless fingerprinting. In *Proceedings of the first ACM Conference on Wireless Network Security*, WiSec '08, pages 46–55, New York, NY, USA, 2008.

[4] P. Eckersley. How unique is your web browser? In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, PETS'10, pages 1–18, Berlin, Heidelberg, 2010. Springer-Verlag.

[5] F. Gont. ICMP Attacks against TCP. RFC 5927 (Informational), July 2010.

[6] F. Guo and T. Chiueh. Sequence number-based MAC address spoof detection. In *Proceedings of the 8th International Conference on Recent Advances in Intrusion Detection*, RAID'05, pages 309–329, Berlin, Heidelberg, 2006.

[7] J. Hall, M. Barbeau, and E. Kranakis. Enhancing intrusion detection in wireless networks using radio frequency fingerprinting. In *Proceedings of the 3rd IASTED International Conference on Communications, Internet and Information Technology (CIIT)*, pages 201–206, 2004.

[8] T. Kohno, A. Broido, and K. C. Claffy. Remote Physical Device Fingerprinting. *IEEE Transactions on Dependable Security Computing*, 2(2):93–108, Apr. 2005.

[9] S. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. In *Proceedings of Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 1 of *INFOCOM'99*, pages 227–234, 1999.

[10] V. Paxson. On calibrating measurements of packet transit times. In *Proceedings of the 1998 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '98/PERFORMANCE '98, pages 11–21, New York, NY, USA, 1998. ACM.

[11] M. Smart, G. R. Malan, and F. Jahanian. Defeating TCP/IP stack fingerprinting. In *Proceedings of the 9th conference on USENIX Security Symposium - Volume 9*, SSYM'00, pages 17–17, Berkeley, CA, USA, 2000. USENIX Association.

[12] G. Taleck. Ambiguity resolution via passive OS fingerprinting. In *Recent Advances in Intrusion Detection*, pages 2003, ISSU 2820, 192–206. Springer, 2003.