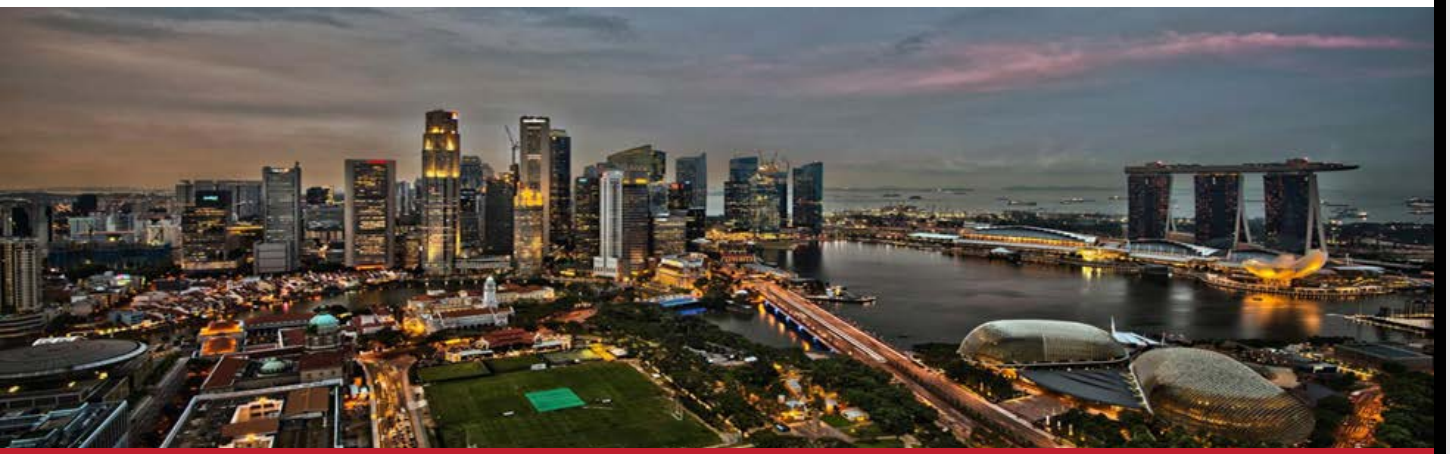
WI-IAT 2015

December 6-9, Singapore

2015 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology

6–9 December 2015, Singapore



Sponsored by

DataSpark clarity redefined (Gold Sponsor)

IEEE

IEEE computer society

Web Intelligence Consortium 2002

acm

SOCIETY

Organized by

Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY)

LARC Living Analytics Research Centre

NANYANG TECHNOLOGICAL UNIVERSITY

SMU Singapore Management University

Supported by

SINGAPORE EXHIBITION & CONVENTION BUREAU

Held in

YourSingapore

Published by

CPS Conference Publishing Services

2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology

WI-IAT 2015

Table of Contents

Volume - 3

The Third International Workshop on Data Oriented Constructive Mining and the Seventh International Workshop on Emergent Intelligence Networked Agents (DOCMAS/WEIN)

International Workshop on Complex Methods for Data and Web Mining (CMDWM)

The Fourth International Workshop on Web Personalization, Recommender Systems and Social Media (WPRSM)

The International Workshop on Knowledge Management of Web Social Media (KMWSM)

The Sixth International Workshop on Intelligent E-government and Emergency Management (EGOVEM)

The Eighth International Workshop on Natural Language Processing on Ontology Engineering (NLPOE)

The International Workshop on Ensemble of Anomaly Detection Algorithms (EADA)

WI-IAT 2015 Poster Papers

WI-IAT 2015 Demo Papers

IEEE Computer Society Order Number E5650
BMS Part Number CFP15WEB-USB
ISBN 978-1-4673-9617-2

Additional copies may be ordered from:

IEEE Computer Society
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314
Tel: + 1 800 272 6657
Fax: + 1 714 821 4641
http://computer.org/cspress
csbooks@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: + 1 732 981 0060
Fax: + 1 732 981 9667
http://shop.ieee.org/store/
customer-service@ieee.org

IEEE Computer Society
Asia/Pacific Office
Watanabe Bldg., 1-4-2
Minami-Aoyama
Minato-ku, Tokyo 107-0062
JAPAN
Tel: + 81 3 3408 3118
Fax: + 81 3 3408 3553
tokyo.ofc@computer.org

Individual paper REPRINTS may be ordered at: <reprints@computer.org>

Editorial production by Lisa O'Conner
Cover art production by Mark Bartosik

IEEE Computer Society
***Conference Publishing Services* (CPS)**
http://www.computer.org/cps

2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology

A Clock Skew Replication Attack Detection Approach Utilizing the Resolution of System Time

Komang Oka Saputra*†, Wei-Chung Teng*, and Yi-Hao Chu*
*Department of Computer Science and Information Engineering
National Taiwan University of Science and Technology, Taipei 106, Taiwan
†Department of Electrical and Computer Engineering, Udayana University, Bali 80361, Indonesia
Email: okasaputra@ee.unud.ac.id, weichung@csie.ntust.edu.tw, M10215046@mail.ntust.edu.tw

***Abstract*—The clock skew, or the physical ticking rate difference between two digital clocks, has been revealed to have potential on serving as the device fingerprint for identification/authentication purpose. However, it remains as an open issue to detect clock skew replication behavior, which is realized by sending altered timestamps. In this study, it is confirmed that an attacker may fake any target skew with the error being no more than 1ppm in local network environments. Besides, it is also observed that the value of fake timestamps are affected by the time resolution of the attacker's system clock. When the resolution is 1 ms or lower, a relatively large jump between consecutive offsets happens regularly, and the scale of each jump is theoretically the very time resolution of the attacker's system clock. This characteristic is thus adopted to develop a filtering method such that the receiver is able to detect fake timestamps. When the periodical jumps are detected, the filter module abandons these jumps to recover the original clock skew. Experimental results on 15.6 ms and 1 ms time resolutions show that the developed method is effective to detect skew replication attacks, and the errors of the recovered clock skews are no more than 1ppm from the real skews of the attackers.**

***Index Terms*—clock skew; replication attack; time resolution**

I. Introduction

One of the essential supporting technologies for the upcoming Internet of Things era is identifying connected devices for services with security concerns. Apart from the cryptographic means, identifiable information like IP address, MAC address, and cookies are vulnerable to replication attack. Alternatively, a physical amount called *clock skew* is revealed to be unique to each device in parts per million (ppm) or higher precision and is thus appropriate to be used in device fingerprinting [1]–[3]. The skew of a digital clock is the frequency difference between the clock and true time, and the clock skew of a device to the measurer is the *relative clock skew* of the device. In this paper, only relative clock skew is discussed.

Although it is hard to alter the ticking rate of a digital clock, it is intuitive to fake a clock skew by slightly adjusting each timestamp such that the monotone-increasing time sequence proceeds in a stable yet different speed. Huang and Teng proposed a method to detect fake timestamps by asking the sender to change the sending period frequently [4]. However, their approach is limited to one-hop communications in wireless sensor networks (WSN) where the transmission delay variation is usually negligible. In a general, multi-hop network, the packet delay variation makes it much harder to determine if the received timestamps are origin ones or not.

In this research, a method is developed to help distinguish the fake clock skews from the real ones. This method utilizes the restriction that the timestamps have to proceed in unit of the system time resolution. According to this restriction, there exists error between the forged time and the target to fake in each timestamp, and the error accumulate as time passes. The sender, or the attacker, has to compensate the errors when the accumulated values become greater than the time resolution, and this behavior causes a periodical *jump* between consecutive timestamps. Since the scales of the jumps are theoretically the resolution of the attacker's system clock, we can use this characteristic to develop a filtering method for the measurer to detect the fake timestamps.

In the following sections, we first demonstrate how an attacker can forge any clock skew by manipulating their timestamps to match the target skew. It is than explained in detail how to detect altered timestamps by detecting regular jumps when the system time resolution of the attacker be 1 ms or lower. Finally, experiments are conducted to verify that the proposed method is effective. The results also show that the recovered clock skews from the faked timestamps are very close (no more than ± 1 ppm error) to the attacker's origin clock skews.

II. How to Fake a Clock Skew

To measure the relative clock skew of a device over network, the measurer M at first collects the timestamps from the target device T. For each received timestamp, an offset is calculated by subtracting the receiving time from the sending time of the packet. Fig. 1 shows the offsets collected in one measurement where the x-*axis* is the measurer's elapsed time after receiving the first packet, and the y-*axis* is offset in unit of microsecond. Each blue circle represents one offset, and consecutive offsets are connected by blue lines. It is obvious that the lower bound of the offset set shapes in a strict line, and the slope of this line is the clock skew of the device. There are many sophisticated approaches [5]–[7] to derive the slope besides linear regression.

978-1-4673-9618-9/15 $31.00 © 2015 IEEE
DOI 10.1109/WI-IAT.2015.10

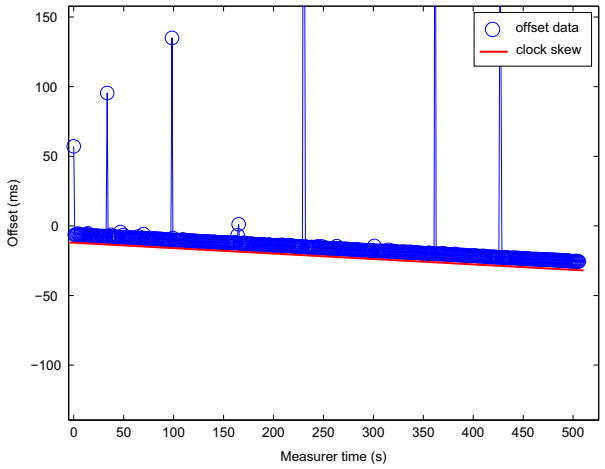


Fig. 1. An offset set and the estimated clock skew from the lower bound.

A. Timestamps and the Sending Interval

Now assume that the attacker A knows the clock skew of T to A: s_{TA}, and A wants to replicate the clock skew of T to cheat M. Every time one second passed in A, $1+s_{AM}$ second passed in M. To make M get $1+s_{TM}$ instead of $1+s_{AM}$, A has to multiply the rate r to the timestamps where

$$r = \frac{1+s_{TM}}{1+s_{AM}}.$$

Let t_1 stands for the first timestamp, or the sending time of the first packet, then the ith altered timestamp t'_i should be

$$t'_i = t_1 + r(t_i - t_1).$$

If A does not know the value of s_{AM}, A can approximate it by the following equation:

$$1 + s_{TM} = (1 + s_{TA})(1 + s_{AM}) \simeq 1 + s_{TA} + s_{AM},$$

or $s_{AM} \simeq s_{TM} - s_{TA}$. Since the absolute values of the clock skews rarely exceed 200 ppm [7], the productions of two clock skews are negligible here.

The weakness of this approach is that M can easily defend from this attack by asking a fixed sending period Δt. Since the difference between two consecutive timestamps t_i and t_{i+1} has to be Δt, A can not modify the value of t_{i+1} without being detected anymore. The alternative way to fake the clock skew is to change the sending period to $r^{-1}\Delta t$.

For example, assume the attacker A knows that s_{TM} is 20 ppm and s_{AM} is 200 ppm, and the measurer M asks for 1 second sending period. It is clear that A's clock ticks faster than T's clock, which is still slightly faster than M's clock. To fake s_{TM}, A sends out timestamps but keep the difference between any two consecutive values 1 second. However, the real sending interval in A's time axis is $1\cdot(1+200\cdot 10^{-6})/(1+20\cdot 10^{-6}) \simeq 1.00018$ seconds.

B. The System Time Resolution Limitation

Due to the computer system design, the above mentioned sending interval has to be in unit of system time resolution. This resolution is determined by the type of operating systems. For instance, it is 1 μs in Linux and Android systems, and the default time resolution in recent Microsoft Windows systems is 15.6 ms. However, users of Windows systems may change the time resolution to up to 1 ms via system calls.

Let k denotes the system time resolution of A. If a process is programed to sleep for Δt seconds to get the next timestamp, the real sleeping time will be $\lceil \frac{\Delta t}{k} \rceil \cdot k$ seconds. For the sake of simplicity, we assume that k divides Δt. Similarly, when A set the sending period to $r^{-1}\Delta t$ seconds, the real sending period becomes $\lceil \frac{r^{-1}\Delta t}{k} \rceil \cdot k$ second, though the fake timestamp only increase Δt seconds each time. The error $e = r^{-1}\Delta t \mod k$ might be tiny, but it accumulates and will eventually affect the estimated clock skew. To compensate this issue, A needs to decrease one tick for every $\lceil \frac{k}{e} \rceil$ rounds.

The following formula summarizes the real sending interval and the fake timestamps of A:

$$t_{i+1} = \begin{cases} t_i + \lceil \frac{r^{-1}\Delta t}{k} - 1 \rceil \cdot k & \text{if } \lceil \frac{k}{e} \rceil \mid i \\ t_i + \lceil \frac{r^{-1}\Delta t}{k} \rceil \cdot k & \text{otherwise} \end{cases} \quad (1)$$

$$t'_{i+1} = t'_i + \Delta t \quad (2)$$

III. Proposed Method to Detecting Clock Skew Replication Attack

According to (1) and (2), there exists a pattern on the timestamps collection at M due to the one tick difference of scale k. Since the offsets on the scatter diagram represent "M's timestamp - A's timestamp", the lessened tick with magnitude of k on A's timestamp will cause the offset jump up or down k. Catching a glimpse on the scatter diagram, the effect by k is not obvious there are also similar jumps that are caused by transmission delay. However, since A has to reduce tick into the timestamps on a fixed interval to make a stable skew, M can spot regular jumps which are totally different comparing with random outliers.

By detecting the time A having attack that is indicated by the occurrence of jumps on the scatter diagram, M can separate A's real timestamps from the timestamps that is addressed for replicating T's skew. Hence, M can recover A's clock skew, and therefore, M can thwart A's attack to keep the system save. The detail of the clock skew replication attack filter can be found in Algorithm 1. Some parameters used in this algorithm are: O is set of all offsets between A and M; m is the threshold set for the maximum offset, or the clock resolution used; $LargerDelay$ is used to store index of the offset-set that is higher than m; $JumpPoint$ is used to store index of the offset-set that is categorized as the position the attack occurs; meanwhile idx is index for the $LargerDelay$ and $JumpPoint$. From lines 3 to 6, Algorithm 1 records into $LargerDelay$ matrix the position in which A is indicated to do attack. Since $LargerDelay$ can be full filed by the position of A's attack and also outliers, Algorithm 1 then separating the

Algorithm 1 Clock skew replication attack filter

Require: $\boldsymbol{O}$, m

1: $LargerDelay$ = $null$
2: $JumpPoint$ = $null$
3: **for** $i = 1; i \leq O.length;\ i++$ **do**
4: **if** $The\ Absolute\ value\ of\ (O_{i+1} - O_i) \geq m$ **then**
5: Recording $(i+1)$ to $LargerDelay$
6: **end if**
7: **end for**
8: **for all** $idx \in LargerDelay$ **do**
9: Finding each continuous idx and then grouping them as $[first\ idx,\ last\ idx]$
10: **if** $first\ idx\ ==\ last\ idx$ **then**
11: Recording idx to $JumpPoint$
12: **end if**
13: **end for**
14: **for all** $idx \in JumpPoint$ **do**
15: O_{temp} = $O_{idx} - O_{idx-1}$
16: **for** $j = idx;\ j \leq JumpPoint.length;\ j++$ **do**
17: O_j = $O_j - O_{temp}$
18: **end for**
19: **end for**

position of A's attack indicated by a non-consecutive index in $LargerDelay$, in which the result is stored to $JumpPoint$ matrix (Lines 8 to 13). Finally, code from line 14 to 19 rebuilds O by removing the effect of A's attack. Since now O no longer contains the timestamps manipulation effect by A, when we estimating the clock skew of O, the result will be just the clock skew of A to M. Therefore, M can identify that A trying to obtain an illegal access to M.

IV. Evaluation Results

Two kinds of experiments are conducted to evaluated the proposed method. A notebook with dual operating systems, Ubuntu 14.04 and Windows 7, was used as A; And a PC with Ubuntu 14.04 operating system plays the role of M.

A. Experiments of Clock Skew Replication

At first, we evaluated the clock skew replication attack method on the 15.6 ms clock resolution. We set the attacker in Windows operating system with default 15.6 ms clock resolution. As the based of the evaluation, Fig. 1 shows the scatter diagram of offsets with the original clock skew s_{AM}. The red line in Fig. 1 is obtained by running the linear programming algorithm [5]. The slope of this line, or the clock skew, is -15.5 ppm.

Three skew values: -215.5 ppm, -35.5 ppm, and -18.5 ppm are the target to be replicated. To reach these targets, on each attack, the attacker has to decrease its speed in the amount of 200 ppm, 20 ppm, and 3 ppm respectively. As an example, Fig. 2 shows the scatter diagram offsets that are obtained when the attacker tried to fake the -215.5 ppm skew target. The measured clock skew in this attack is -215.79 ppm.

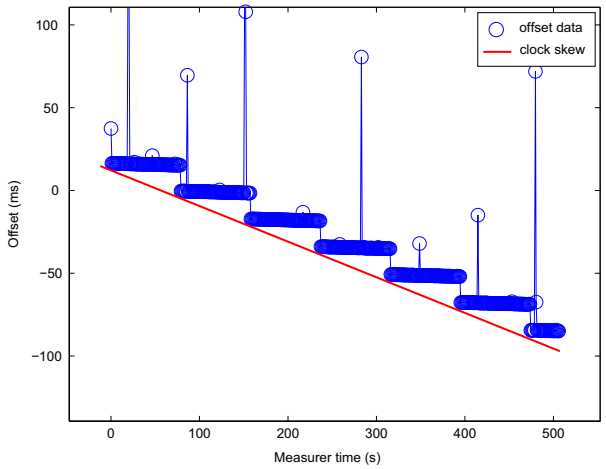


Fig. 2. Offsets when the attacker with 15.6 ms resolution imitating -215.5 ppm. The estimated clock skew by using LPA is -215.79 ppm.

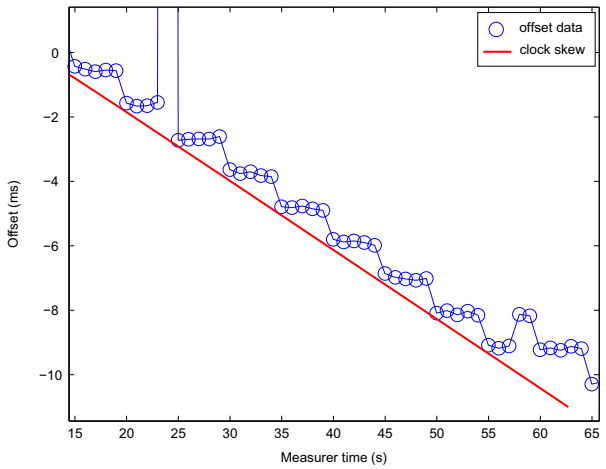


Fig. 3. Part of offsets when the attacker with 1 ms resolution imitating -215.5 ppm. The estimated clock skew by using LPA is -214.83 ppm.

From Fig. 2 we can also observe that the densest part of of the offsets on the scatter diagram is broken up due to the presence of jump points. Furthermore, we found that the magnitude of each jump is nearly 15.6 ms, or it is close to the value of the clock resolution used by the attacker.

Similarly, we repeated experiments in 1 ms time resolution. Fig. 3 shows the result when A tries to imitate the -215.5 ppm skew. An accurate replication attack is observed with only 0.67 ppm error.

Table I details the results of all the above experiments. Some notations used here are: W for Windows, L for Ubuntu 14.04, 15m for the 15 ms time resolution used by the attacker, 1m for the 1 ms, 1μ for the 1 μs, 200p for the condition when attacker trying to add 200 ppm in its original skew, 20p when the attacker trying to add 20 ppm, and 3p when the attacker

TABLE I
RESULTS OF THE CLOCK SKEW REPLICATION ATTACK METHOD

Experimental combinations	Original clock skew (ppm)	Clock skew target (ppm)	Clock skew result (ppm)	Error (ppm)
W15m200p	-15.5	-215.5	-215.79	0.29
W1m200p			-214.83	0.67
L1μ200p			-214.96	0.54
W15m20p		-35.5	-35.33	0.17
W1m20p			-35.37	0.13
L1μ20p			-35.48	0.12
W15m3p		-18.5	-18.69	0.19
W1m3p			-18.12	0.38
L1μ3p			-18.58	0.08

TABLE II
RESULTS OF THE CLOCK SKEW REPLICATION ATTACK FILTER

Experimental combinations	Detected skew (ppm)	Recovered skew (ppm)	$\lvert Recovered - Original \rvert$ (ppm)
W15m200p	-215.79	-15.53	0.03
W15m20p	-35.33	-15.21	0.29
W15m3p	-18.69	-15.57	0.3
W1m200p	-214.83	-16.07	0.57
W1m20p	-35.37	-15.91	0.41
W1m3p	-18.12	-14.97	0.53

trying to add 3 ppm. It is obvious from Table I that even when the notebook attacker using Windows (1 ms resolution) or Linux (1 μs resolution), its replication attacks are finish with accurate results (all the errors are lower than 1 ppm).

The period the jump point occurs on 15.6 ms and 1 ms resolutions are different. By observing Figs. 4 and 5 more detail, we can find that the jump point occurs every 78 seconds and 5 seconds for the 15.6 ms and 1 ms resolutions respectively. The longer the measurement the more the jump point will occur. Hence, to obtain information about the relation between the number of jump point with the skew to be faked, we tried to arrange the length of the measurement, from 30 seconds into maximum 1000 seconds, and to try any possible skew values to be imitated.

B. Experiments of Filtering the Replication Attack

As the jump point phenomenon only occurs on Windows with millisecond clock resolutions, we evaluated the proposed clock skew replication filter only for the 15.6 ms and 1 ms clock resolutions. With similar notations as used in Table I, Table II summarizes all the filtering results. On all the experiment combinations, the clock skew replication filter could recover the clock skews of the attacker into its original skew. Comparing with the original clock skew of -15.5 ppm, the highest error is only 0.57 ppm when the attacker try to imitate -215.5 ppm with clock resolution of 1ms (W1m200p).

As an illustration for the filtering process by the measurer, we provide Figs. 4 (for W15m200p). From this figure we can observe that new offset values are produced by the filter method, indicated by the * symbol on the scatter diagram. From these recovered offsets a new clock skew is produced, in which this clock skew is the genuine clock skew of the attacker.

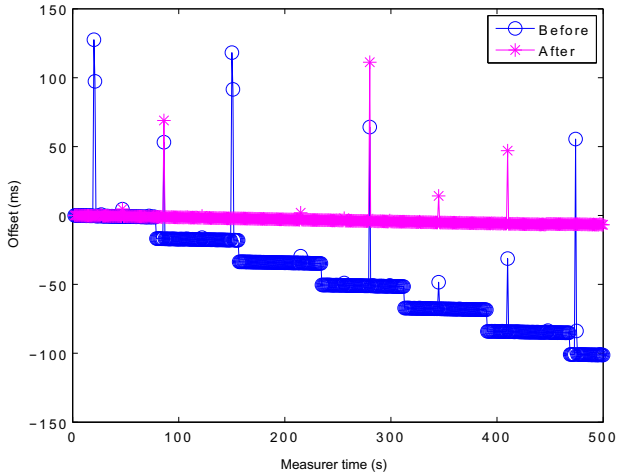


Fig. 4. Result of the skew replication attack filter. Original skew (blue offset) is -215.79 ppm. Skew of the recovered offset is -15.53 ppm.

V. CONCLUSION

In this paper, we proposed a filtering method to defend from clock skew replication attack by detecting the unnatural jumps in fake timestamps. Implementation on the Windows system also demonstrated the effectiveness of the proposed method. Although the proposed method may become ineffective to devices with 1 μs or higher resolutions, it provides a hint on designing clock skew measuring scheme.

ACKNOWLEDGMENT

This work was supported by the Ministry of Science and Technology under Grant MOST 104-2923-E-011-005-MY3 and MOST 104-2221-E-011-070.

REFERENCES

[1] T. Kohno, A. Broido, and K. Claffy, "Remote physical device fingerprinting," *IEEE Trans. Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, 2005.

[2] S. Sharma, A. Hussain, and H. Saran, "Experience with heterogenous clock-skew based device fingerprinting," in *Proc. 2012 Workshop on Learning from Authoritative Security Experiment Results*, 2012, pp. 9–18.

[3] M. Cristea and B. Groza, "Fingerprinting smartphones remotely via ICMP timestamps," *IEEE Commun. Lett.*, vol. 17, no. 6, pp. 1081–1083, Jun. 2013.

[4] D.-J. Huang and W.-C. Teng, "A defense against clock skew replication attacks in wireless sensor networks," *J. Network and Comput. Applicat.*, vol. 39, pp. 26–37, Mar. 2014.

[5] S. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," in *Proc. INFOCOM Conf.*, 1999, pp. 227–234.

[6] M. Aoki, E. Oki, and R. Rojas-Cessa, "Measurement scheme for one-way delay variation with detection and removal of clock skew," *ETRI J.*, vol. 32, no. 6, pp. 854–862, Dec. 2010.

[7] K. Oka Saputra, W.-C. Teng, and T.-H. Chen, "Hough transform-based clock skew measurement over network," *IEEE Trans. Instrum. Meas.*, 2015, published online, to appear in print.