2012 26th IEEE International Conference on Advanced Information Networking and Applications

Clock Skew Based Client Device Identification in Cloud Environments

Ding-Jie Huang *, Kai-Ting Yang *, Chien-Chun Ni †, Wei-Chung Teng *, Tien-Ruey Hsiang *, and Yuh-Jye Lee *
* Department of Computer Science and Information Engineering
National Taiwan University of Science and Technology, Taipei City, Taiwan 106
Email: {D9515002, D9615007, weichung, trhsiang, yuh-jye}@mail.ntust.edu.tw
† Department of Computer Science
Stony Brook University, Stony Brook, NY 11794, USA
Email: chni@cs.stonybrook.edu

***Abstract*—Along with the growth of cloud computing and mobile devices, the importance of client device identity concern over cloud environment is emerging. To provide a lightweight yet reliable method for device identification, an application layer approach based on clock skew fingerprint is proposed. The developed experimental platform adapts AJAX technology to collect the timestamps of client devices in the cloud server during connection time, then calculate the clock skews of client devices. Few methods based on linear regression and piecewise minimum algorithm are developed to optimize the precision and shorten timestamp collection process. A jump point detection scheme is also proposed to resolve the offset drifting problem, which is usually caused by switching network or temporary disconnection. Finally, two experiments are conducted to study the effectiveness of clock skew fingerprint, and the results illustrate that the false positive rate and the false negative rate, in the worst case, are both no more than 8% when the tolerance threshold is set appropriately.**

***Index Terms*—clock skew, device identity, cloud service, jump point detection**

I. Introduction

The growth of cloud-based services in recent years has significantly changed the way how people use computers and mobile devices, and also the way security attacks may be launched. The architecture of cloud differs from classic computer network in many ways. Network connections become part of infrastructure on basic operations like data accessing, and the server threats are further isolated under virtualization technologies. Therefore, classic security issues such as user privacy, data confidentiality and regulation concerns should be reconsidered under the cloud computing environments.

This paper studies the client device identification problem of cloud security. Nowadays, people often subscribe cloud services through personal devices such as mobile phones, tablets, and laptop computers. Therefore, user identity can be associated to dedicated hardware. Device identification in the cloud is useful in detecting unauthorized account access and locating stolen devices.

Device identification is realized by maintaining a registered list of physical devices which are associated with valid users. Once a user logs into the service via an unregistered device, service provider may ask the user to pass additional verification, or raise an alarm to the system supervisor for possible invalid user login. Many candidates can be used as the fingerprint of physical devices, such as IP address, MAC address, cookie, or web browser's configurations [1]. However, these attributes suffer the weaknesses of easy to forge, lack of uniqueness, and varying with environments, which make them insufficient to serve as device fingerprinting. In this paper, we propose clock skew, a distinctive fingerprint, to identify different devices.

Clock skew is the difference of clocking speed between two clocks. Generally, modern processor's clocks present two properties [2]–[4]: first, the clock skew between two devices is relatively stable over time; second, there is distinguishable clock skew between any two physical devices. According to these two properties, clock skew can be regarded as a fingerprint of any device with digital clock. Clock skew has been widely used as an attack method to reveal a web host behind HoneyPot or Tor networks [4]–[6]. Later, it is also used to verify the identity of sensor motes [7,8]. This work applies clock skew to identify client devices for server security in cloud environments, or even in classic web systems.

In this paper, we treat clock skew from a novel aspect and regard it as a detection mechanism of adversary. Also, to display the usability of clock skew, a web page based skew detection system is constructed to collect the timestamps, and five different methods are implemented on it to calculate the clock skews of client devices. We provide an AJAX-based technique to periodically collect time information from cloud service subscribers to the server. Device fingerprinting is then performed by estimating the clock skew via this time information. Through experiments performed on a testing platform containing over 100 devices, the clock skew effectively identifies client devices regardless of underlying networking channels such as 3G, Wi-Fi, or ADSL, etc.

Furthermore, during our experiments, we observed that sometimes packet offsets drift dramatically and thus cause a *jump point*. This problem is mainly caused by global time synchronization or network connection hand-off. In previous work, since the packet collection period is long, this phenomenon is ignored and have not been solved. To minimized the packet collection period and speed up the clock skew computation, a jump point detection scheme is provided.

1550-445X/12 $26.00 © 2012 IEEE
DOI 10.1109/AINA.2012.51

In this work, we introduce clock skew as a effective client device identity in cloud services, and provide a solution to the jump point problem which to our knowledge has not been discussed before.

II. Related Work

At first, we review several modern host identification schemes. Then, clock skew based techniques developed for wireless sensor networks and the Internet are discussed.

A. Host Identification Techniques

In previous studies [9], once an adversary successfully forge its identity to infect the cloud server, the adversary might prompt serious damage either to the service provider or to the client. Thus, an effective scheme to identifying different clients is essential for any SaaS based service.

On the other hand, Remote host identification or fingerprinting has been studied broadly. Possible applications vary from Bluetooth signal [10] to RFID tags. Gerdes et al. [11] proposed a method to uniquely identify Ethernet devices by analyzing their analog signals. Later, Rasmussen et al. [12] extended the radio fingerprinting technology to wireless sensor networks, and demonstrated its feasibility through experiments. Eckersley [1] presented a web browser-based device fingerprinting approach which uses browser's configurations and plug-in information to characterize the device. Eckersley's also showed that web browser's information can be easily traced.

B. Clock Skew based Attack and Defense Schemes

Clock skew has been widely used in various attacks to detect and to reveal hidden hosts. In [4], Kohno et al. exploited clock skew to fingerprint a remote physical device by stealthily record and analyze its ICMP or TCP timestamps. However, for real world applications, using ICMP and TCP timestamps have their limitation. ICMP timestamps are blocked by many firewalls, and some operating systems in default disable TCP timestamps. Furthermore, their approach failed in anonymous networks like Tor in which end-to-end TCP connection is not possible [6].

Murdoch [5] proposed a clock skew based attack to reveal hidden services. The pseudonymous server identity was revealed due to the shift of clock skew which results from increased server load and accordingly the CPU temperature. This attack works in Tor network. However, it highly relies on large amount of traffic from the hidden server through Tor network. Also, it requires large amount of timestamps in a short period of time to perform adequate clock-skew estimation.

Later Zander and Murdoch developed an enhanced attack with synchronized sampling technique [6] which significantly reduces the quantization error and thus cut the heavy network traffic necessary to previous attack. Also, their work is the first one that undertakes the clock skew estimation through HTTP protocol. However, this attack model can not be directly applied to server side for defense purpose. We discuss more about the reason later at section III.

All of the above clock skew based techniques are attacking methods that probe service providers on the Internet. To deploy clock skew as a defense mechanism, Huang et al. [7] proposed a method to utilize clock skew on node identification in wireless sensor networks. In their paper, clock skew fingerprinting is proposed as a countermeasure against Sybil attack. Uddin's work [8] also verified that the above-mentioned skew-based scheme an effective fingerprinting approach if under a covert channel.

III. Clock Skew Based Host Identification: Scenario and Preliminaries

In this section, the scenario and preliminaries of clock skew based host identification are introduced. In the first part, a scenario of how clock skew based scheme assists device identification and malicious adversary detection is presented. The second part discusses preliminary basic knowledge of computing clock skews and corresponding terminology. Then, a brief description of the experimental platform is given. In the last part, the time collection process is further analyzed.

A. Scenario of Clock Skew Based Client Device Identification

1) Device identification system construction: The scenario of the client device identification system is showed in Fig. 1. Consider a user trying to login to a web-based cloud server. To confirm the validity of this login, the server checks if this user already has one or more registered devices in the skew value database.

If not, the server performs secondary authentication on the client. Some currently popular methods include cell phone verification, email verification, and interactive method, to name but a few. If the user cannot pass the verification, login is denied. Users who passed the identification process may choose whether to register the current device or not. In the former case, a timestamp collection server starts to monitor the time difference between itself and the client device, and calculates the relative clock skew accordingly. The estimated clock skew value become the fingerprint of the client device, and is stored in a database for later login.

On the contrary, if there exist registered device in the database, the server compares the clock skew of current client device with the registered one. If the difference of these two skews is under a tolerance threshold, the server accepts the device and consider the client has passed the verification. In contrast, if the difference exceeds the threshold, this account might be under an account hijacking attack. In this case, the server then requires the user to provide further information to verify his/her identity. The following steps are the same with the no registered client device case. Finally, the server should raise an alarm to notify the associated client when potential malicious intention is detected.

2) Timestamp collection system of a cloud service: With the aid of clock skew fingerprinting technique, a cloud service gains additional protection against malicious attackers. In

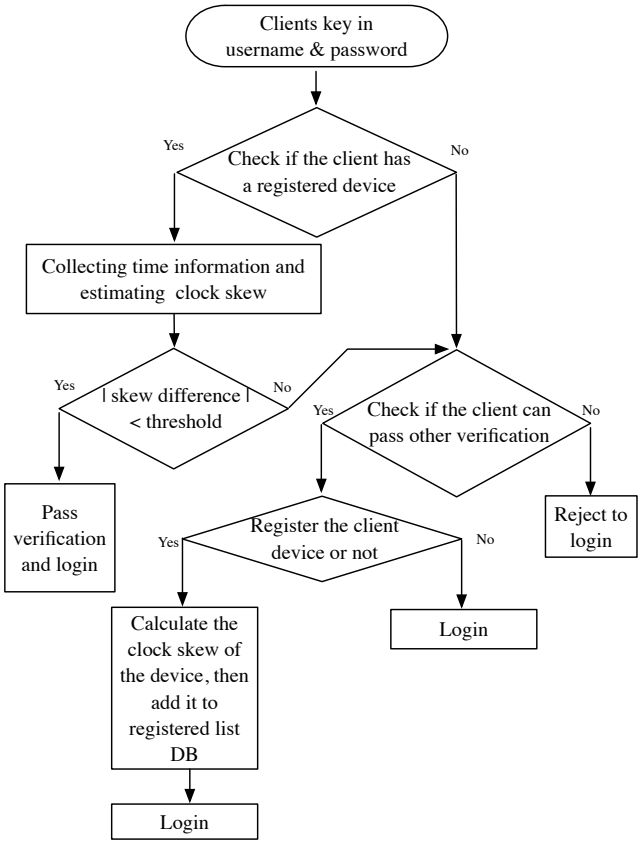


Fig. 1. Flowchart of clock skew based host identification system.

fact, device fingerprint, as a physical attribute, is superior to network parameters like IP address, MAC address, and cookie. Moreover, clock skew's two major properties, uneasy to forge and device distinctness, solid our method as a prospect candidate for cloud-based applications.

As stated earlier, Zander and Murdoch provided a clock skew estimating method based on HTTP request [6]. However, their method cannot be directly applied at the server side in the above scenario. This is because their approach must frequently asks for the time information of remote host through HTTP request, which is infeasible for a web server to perform in the same way. Therefore, a cloud server needs a different approach to obtain time information from a client.

To force the client to return its own time informations back to server, AJAX technique is applied in our system because every packet generated by AJAX contains the corresponding timestamp. Since AJAX is implemented by Javascript and Javascript is generally supported by web browsers, we assume that the execution of Javascript is allowed.

The architecture of the timestamp collection system is illustrated in Fig. 2. On the client side, a client connects to web server through Secure Sockets Layer (SSL) to provide a secured channel. On the server side, the main web server provides general services to clients. Meanwhile, a timestamp collection server is settled to gather timestamps from clients.

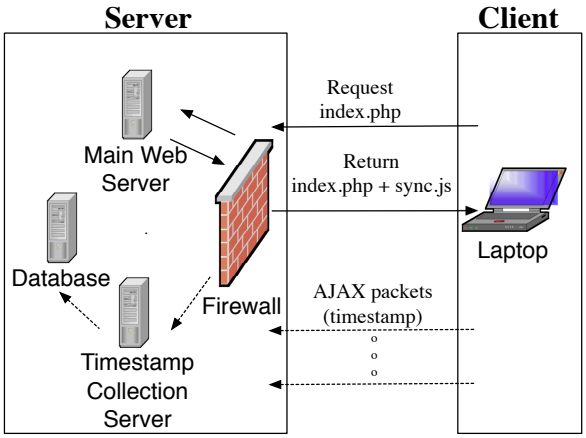


Fig. 2. Scenario of timestamp collection.

The calculated clock skew are stored in the database server for future use. Once a client requests the index page from the main web server, it returns the page including a script sync.js. This script then asks the client device to periodically send back its time information to the timestamp collection server.

In fact, we have built a prototype system to perform experiments of clock skew identification following the above scenario. The main web server equips Ubuntu server operating system, running Apache web server. Because our skew estimating method is based on Javascript, the accuracy of timestamps is limited in one millisecond.

B. Terminology of Clock Behavior

The terminology used to represent the clock characteristics is as follows. The nomenclature from [3,8,13] is adapted is this paper. Consider $C_x(t)$ as the time reported by the clock of device x at real time t, $C'_x(t) \equiv dC_x(t)/dt$ and $C''_x(t) \equiv d^2C_x(t)/dt^2$, $\forall t \geq 0$. Let C_c and C_s be the clocks of client and the timestamp collection server respectively:

1) **Offset:** The difference between the time reported by C_c and by C_s, e.g., the offset of the client clock C_c relative to the server clock C_s is $C_c(t) - C_s(t)$, $\forall t \geq 0$.
2) **Frequency:** The rate at which the clock progresses, e.g., the frequency at time t of C_c is $C'_c(t)$.
3) **Skew (δ):** The difference in the frequencies of two clocks, e.g., the skew of C_c relative to C_s at time t is $\delta(t) = C'_c(t) - C'_s(t)$.
4) **Drift:** The drift of C_c relative to C_s at time t is $C''_c(t) - C''_s(t)$.

According to the above definitions, if the server accumulates sufficient client time information, the clock skew δ of this client can be computed by the server locally.

C. Usage of Timestamps

With AJAX, timestamps can be collected by server in every packet returned from client. Unless further specified, all clock skew estimation is based on server clock C_s. Assume that the timestamp collection server has received n AJAX packets

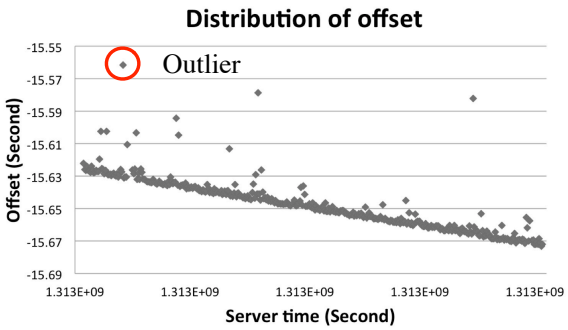


Fig. 3. Distribution of offset collected from the timestamp collection server.

from a client. Let timestamp t_i^c represents the C_c time when i^{th} packet sent out by the client; similarly, timestamp t_i^s denotes the C_s time when the i^{th} packet is received by the server. The estimated offset between server and client for the i^{th} packet is denoted as o_i, where $o_i = t_i^s - t_i^c$. Also, the period between the i^{th} packet and the j^{th} packet according to the server's clock C_s is denoted as x_{ij}, where $x_{ij} = t_j^s - t_i^s$.

To illustrate the relation between server time and packet offset more clearly, (t_i^s, o_i) is plotted into a scatter diagram; furthermore, the clock skew δ between server and client can be estimated as the slope of this diagram. An example is shown in Fig. 3 , where trend of these dataset is decreasing with a negative slope, which means that the clock skew between client and server is negative.

IV. Clock Skew Estimation

As shown in previous section, skew can be estimated as the slope of the scatter diagram. However, due to occasionally network delay or jitter, there are some outlier nodes that can not be used to compute the clock skew. In this section, two basic techniques are illustrated to estimate the clock skew between the timestamp collection server and the client: linear regression and piecewise minimum algorithm. Moreover, for clock skew based host identification, we implement five methods to estimate the clock skew between the server and the client, and analyze the performance of each estimation.

A. Linear Regression Algorithm

Linear regression is a method for a set of data points to approach one line. Although this method is not robust while significant outliers exist in the data set [3], it is easy to implement, and with less computation overhead. Here, four different types of mechanism based on linear regression are proposed to approach the clock skew. In the following paragraph, $LR(N_{ij})$ denotes the linear regression calculation for data set N_{ij}, which contains the data from (t_i^s, o_i) to (t_j^s, o_j).

1) Accumulated Skew: For accumulated skew, while packets sent from the client are received by the server, the server computes the estimated skew immediately. The estimated skew can be represented as $LR(N_{1i})$, while receiving i^{th} request from the client.

In accumulated skew, every data, even outlier, is accumulated in the data set. With vast data and time, this method provides stable and reliable result. However, in a short period of time, accumulated skew is dramatically influenced by outliers. Moreover, errors caused by the outliers continuously affect the result for all the following computation. For example, as shown in Fig. 4(a), estimated skews fluctuate enormously at the red circle part due to huge network delay at this time.

2) Skew with Sliding-Windows: Comparing with the accumulated skew, a sliding-window skew takes sampling only from the most recently short period of time. This prevent the largely fluctuated data from poisoning skew estimation in long term.

For sampling windows with size w, the sliding-windows skew $LR(N_{ij})$ must satisfy $j - i = w$.

Fig. 4(b) shows the result of sliding-windows skew with window size 200; the suggestion on the window size is provided in section V.

As the result indicates, sliding-windows reduced the effect of outliers which is outside the sliding window, but if outliers exist inside the window, this method still suffer from the outlier effect. Therefore, to filter out these outliers, a method to choose the suitable inliers is needed, especially under the environment that huge network delay happens randomly.

Digressively, one may notice that the skew estimation in Fig. 4(b) are all 0 in the beginning. This is because the window size w is 200, the skew estimation before 200 are marked as 0 for no enough data. All sliding-windows based methods have this restriction.

3) Sliding-Windows Skew with Lower-Bound Filter: To disassemble the effect caused by outliers, the most effective method is to filter them out, so a lower bound approach would be helpful here. For example, huge network delay exists at red circle part in Fig. 3 (i.e. the outliers); these points mostly caused by network delay may affect the correctness of skew estimation enormously. In contrast, the offsets lie in the lower part are relative smooth. Hence, the closest skew estimation is calculated from the lower-bound dataset, as previous research did [3]. In [3], Moon et al. suggests piecewise minimum algorithm as a simple and efficient method for estimating clock skew. Since piecewise minimum can be used to extract the lower bound of offset, the algorithm is suitable to serve as the low-bound filter.

For sliding-windows skew with lower-bound filter, the local minimum offset is picked for every m packets in each sliding window w. Thus, the amount of sampling data for skew estimation is reduced to $\lfloor w/m \rfloor$. With this lower-bound filter, only packets with minimal network delay are collected, thereby minimizing the influence of the network. This lower-bound skew estimation can be denoted as $LR(Min(N_{ij}))$, where $Min(N_{ij})$ is the data set of local minimum offset between i^{th} request and j^{th}.

As shown in Fig. 4(c), the skew estimation is much smoother than the one in Fig. 4(b); the window size w is

200, m is 5 in this example, so the total data for one skew estimation is 40. By this method, the clock skew estimation hence stable and capable for reducing the effect of huge network delay.

4) Accumulated Sliding-Windows Skew with Lower-Bound Filter: Since the local minimum offset is useful to find the lower-bound skew, we further calculate the accumulated skews with these local minimum dataset and plot the corresponding skew distribution diagram in Fig. 4(d). We found that this method can both reduce the effect of huge network delay and converge rapidly within 20 packets. This skew estimation can be denoted as $LR(Min(N_{1i}))$ while receiving i^{th} request from the client.

As a result, by comparing these four types of skew estimations, the lower-bound filter has the best performance on filtering out the huge network delay in our experiments.

B. Quick Piecewise Minimum Algorithm

The quick piecewise minimum algorithm is achieved by separating data into n segments and picking two minimum offsets from first segment and fourth segment respectively. By connecting these two minimum offsets, a slope of this line can be obtained. In our experiment, n is set to 4. As shown in Fig. 5, the black circles are offsets, the red squares are the corresponding segments, and the two blue squares are the two local minimum offsets. Thus, the estimated skew is the slope of the black line between two blue squares.

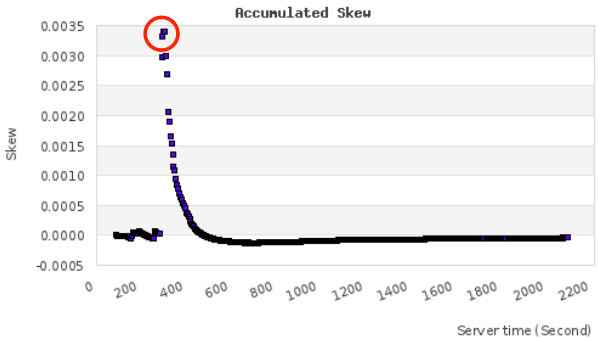
As long as the data set is stable enough, the quick piecewise minimum algorithm can achieve high stability with little computation. Fig. 6 illustrates the skew estimated by this method based on the same data with Fig. 3. Since the quick piecewise minimum algorithm has low computation overhead, it can be efficiently apply to other cloud service.

C. Jump Point Detection and Elimination

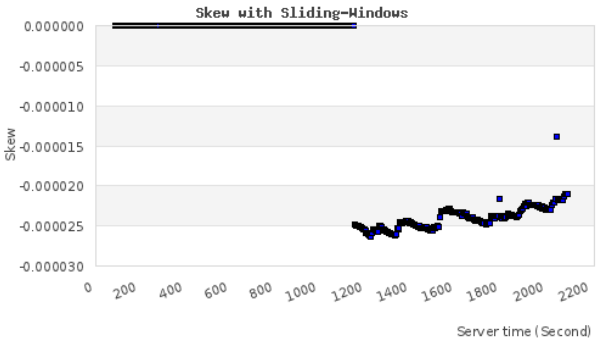
During packet collection period, a *jump point* of offset occurs if the client is performing time synchronization with a time server or roaming between different network providers. An example of jump point, as shown in Fig. 7, is marked by red circle. If this jump point phenomenon does not detected by server, the skew estimation process would be affected by the mass offset error, reacting in an inaccurate result. Thus, an approach to detect jump point occurrence is needed while implementing the skew estimation.

The main idea of this algorithm is to divide sequential timestamps into groups, where every pair of consecutive timestamps in a group produces only a small time difference. Suppose *diff* represents the difference between two adjoining offsets; the *diff* is denoted as d_{ij}, where $d_{ij} = o_j - o_i, \forall i \neq j, i < j$. Also, another threshold k is set to check if the *diff* is too large, thereby detecting the jump point. The basic jump point detection algorithm is as follows:

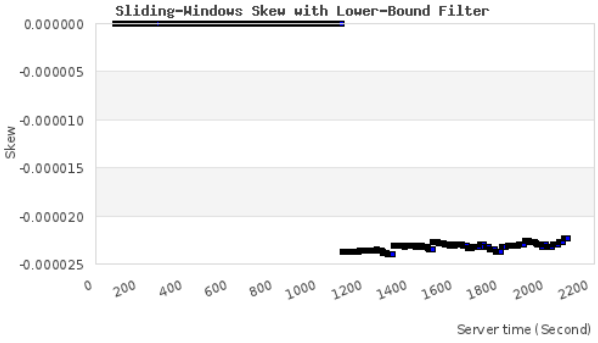
1) Pick the local minimum offsets for every p packets.
2) Compute the *diff* between every pairs of contiguous local minimum offsets.
3) Following detection process is divided into two classes:



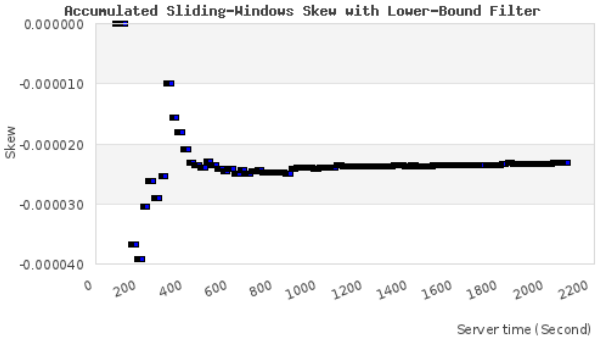
(a) Accumulated Skew



(b) Skew with Sliding-Windows



(c) Sliding-Windows Skew with Lower-Bound Filter



(d) Accumulated Sliding-Windows Skew with Lower-Bound Filter

Fig. 4. Comparison of each skew estimation by applying linear regression.

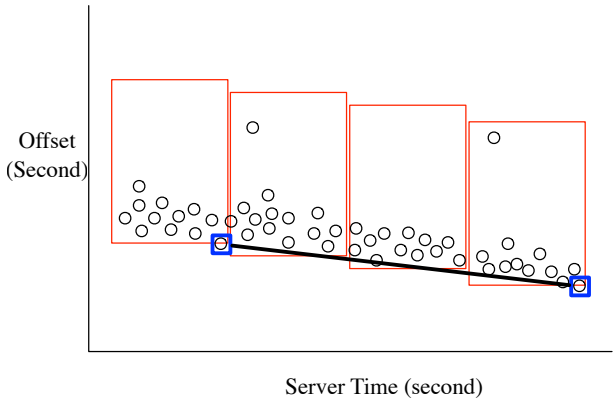


Fig. 5. Implementation of quick piecewise minimum algorithm.

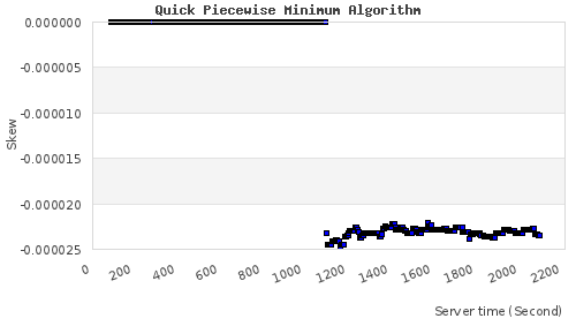


Fig. 6. Skew estimation by applying quick piecewise minimum algorithm.

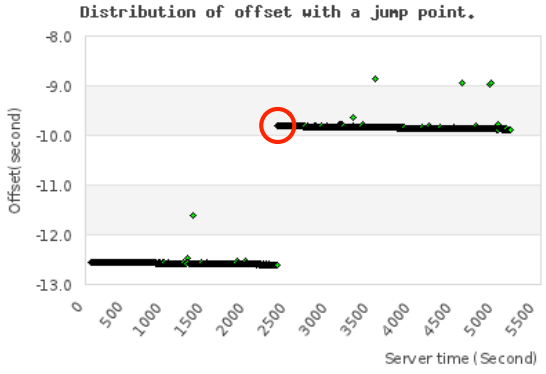


Fig. 7. Offset distribution diagram between server and client with a jump point.

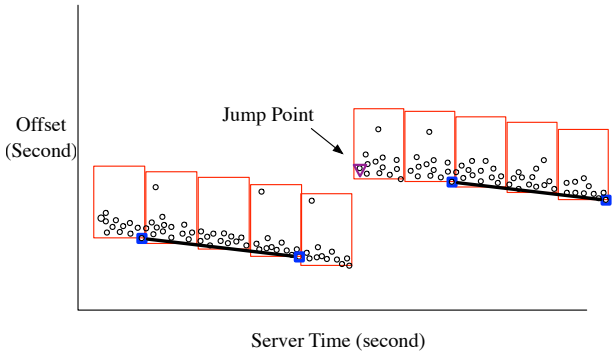


Fig. 8. Detection of jump point between two segments.

a) If derived *diffs* are all positive or all negative:
   - Denote the median of derived *diffs* as $Med(diff)$.
   - If there exists a *diff* that $diff > k \cdot Med(diff)$, a jump point exists inside these p packets.

b) If only part of derived *diffs* are positive:
   - If positive *diffs* are followed by one negative *diff* at x, this x is the jump point.
   - Similarly, negative *diffs* followed by one positive *diff* is processed vice versa.

To prevent the error caused by the jump point, the skew estimation is executed before and after the jump point separately by quick piecewise minimum algorithm. As shown in Fig. 8, while the jump point is located at j^{th} segment, the first half estimated skew is calculated from 1^{th} to $(j-2)^{th}$ segment; the second half is calculated from $(j+1)^{th}$ to the last segment. Two segments are omitted from skew estimation because the jump point may locate at $(j-1)^{th}$ or j^{th} segment. By this segmental method, all candidate skews can be calculated. To represent the moderate value of all skews, we pick the median of these candidate skews as the estimated skew for the corresponding client.

A homogeneous case of jump point is a time gap, which is a period of time of blank packets, as shown in Fig. 9. A time gap generally arises when user is moving around, causing the change of network connection, resulting in a period of disappearance. To detect this kind of jump point, simply checking x_{ij}, where $j = i + 1$. If x_{ij} is larger than twice of the assigned AJAX packet sending period, it is considered as a time gap, and is treated as a jump point as well.

V. Experiments and Evaluation

To evaluate the proposed host identify method, two different experiments are held in this section. Principal parameters of our experiments are set as follows.

To obtain high precision offsets, at least 200 AJAX packets are collected for skew estimation in every experiment. Since the time resolution of Javascript is 1 ms, at least 1000 seconds is required to detect a offset up to 1 microsecond resolution. Since our AJAX script generates a packet for every 5 seconds, the minimum number of packets required would be $1000/5$, or 200.

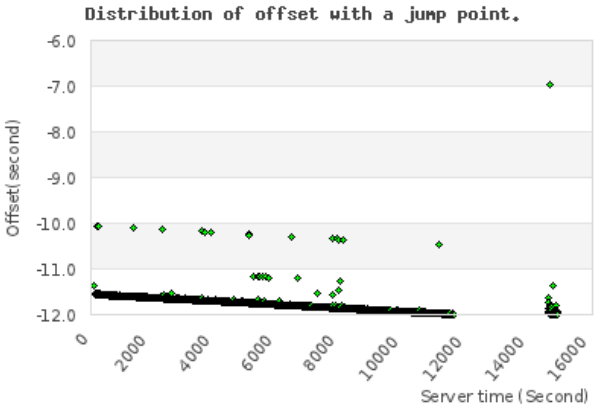


Fig. 9. Offset distribution diagram between server and client with network disconnection.

TABLE I
THE ESTIMATED SKEW OF THE SAME DEVICE UNDER DIFFERENT NETWORK ENVIRONMENTS

Network type	Skew estimation	Packets	No. of IP addr.
LAN	-21.91 ppm	1001	1
	-23.24 ppm	207	1
	-22.74 ppm	13322	1
ADSL	-21.48 ppm	5837	1
	-21.08 ppm	1400	1
3G	-23.24 ppm	951	1
	-23.71 ppm	1027	1
Wi-Fi	-21.79 ppm	9810	1
	-23.06 ppm	1470	1
Tor	-22.53 ppm	15007	55
	-23.22 ppm	12922	57
	-22.88 ppm	24120	108
VM	-113.19 ppm	868	1
	-114.22 ppm	1001	1
	-6.40 ppm	1001	1
	-6.83 ppm	890	1

For the jump point detection, the parameter p is set to 20, and the threshold k is also set to 20. p and k are adjustable according to different network environments; with parameters of smaller values, the jump point detecting process leads to a much sensitive result.

A. Same Device under Different Networks

The first experiment is performed with a laptop connecting to our timestamp collection system via multiple network access techniques. Host behind Tor network and virtual machine (VM) are included in this experiment as well for reference. The purpose of this experiment is to measure the skew variations of one device under different network environments, thereby simulating the situations that a device logins the cloud service from various kind of networks. In this experiment, the device is an Apple MacBook with a 2.4 GHz Intel Core 2 Duo processor, running MAC OS X 10.6.8, and the estimated skew is calculated by jump point detection with quick piecewise minimum algorithm. As shown in table I, we at first connect this laptop to the server via common networks such as LAN, ADSL, 3G, and Wi-Fi.

According to the first four results, it justifies that clock skew estimation is relative independent regardless of network access media. It is worthy to notice that the estimated skew is fluctuated within 2.16 ppm, from -21.08 ppm to -23.24 ppm. Since the clock skew may fluctuate with temperature mentioned in [5,6,8], these experimental results seems acceptable considering the short timestamp collecting time and possible noise related to network environments.

With the skew fluctuation, it is a trend-off to set a reasonable tolerance threshold for detecting malicious login. If the threshold is too small, the false positive rate would be unacceptably high, which means that one device might be treated as different device quite often. On the contrary, if the threshold is too large, the false negative rate would raise, which leads to high probability of accepting unregistered devices. Thus, after analyzing the table I, we consider a threshold of ± 1 ppm appropriate at the present stage. This threshold derives a probability of misjudgement 7.4%, or $(2.16 - 2)/2.16$, assuming the probability density function of skew fluctuation is uniformly distributed.

Furthermore, the experimental environment in Tor is also listed in table I. In our experimental results, 200 packets is not enough to calculate a stable estimation. However, the estimated skews diverge and are close to the common network cases, as long as the collecting time is long enough. Finally, for client hosts running inside a VM, the skews are stable but unpredictably different from the real one. Kohno et al. [4] had indicated that virtual machines do not have constant clock skews, and our experiments also shows the same results. In addition, we found that the estimated skew under a VM is relatively stable without rebooting it; however, after restarting the VM, the skew randomly changes to another stable value. As shown in table I, the estimated skews of two clients are -113.19 ppm and -114.22 ppm respectively, but their skews changed to -6.4 ppm and -6.83 ppm after rebooting the system.

B. Skew Distribution of Different Devices

To study the distinguishable property of clock skew, we collected clock skews of 100 devices estimated by the same timestamp collection server. The results vary from 67 ppm to -499 ppm, and only the most close 90 skews, sorted by their values, are illustrated in Fig. 10. Each cross in this figure represents one device and the y coordinate stands for its clock skew. Each clock skew is bound by an interval of ± 1 ppm range. As shown in this figure, many devices have their intervals overlapped with others, which means that there exists not negligible probability a user may pass the identification test with unregistered devices. The worst case happens at device number 20, whose interval overlaps with other 8 devices. Therefore, with the threshold of 1 ppm, the maximum false negative rate is currently 8%. We believe that further analysis

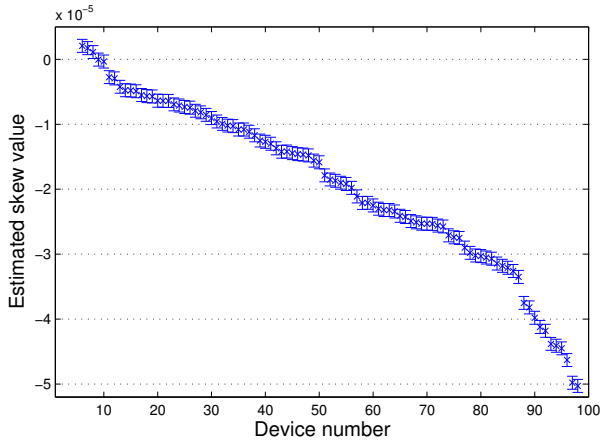


Fig. 10. The sorted skew of estimated devices

on skew estimation of different network environments would help reducing the tolerance threshold, thus reduce both false positive rate and false negative rate.

VI. Conclusions and Future Work

This paper addressed the client device identification issue in cloud environment, and proposed a clock skew based fingerprinting technique and a practical scenario. Client device identification strengthen the account safety, and the proposed scenario suggests a potential secondary authentication approach without users' awareness most of the time. Several classical methods are introduced and implemented to estimate the skews of client devices, and the treatment of jump points, usually caused by switching network or temporary disconnection, is also discussed at the first time. To examine the effectiveness of clock skew, we implemented a web based skew estimation platform and conducted two experiments. The initial study includes over 100 client devices from 5 different network media. The experiment results showed that the false positive rate and the false negative rate, in the worst case, are both no more than 8% when the tolerance threshold is set to ± 1 ppm. With the potential of clock skew fingerprint be revealed, the further work would include improving the precision of skew estimation utilizing linear programming method and accumulating the knowledge of skew fluctuation in different network media.

Acknowledgment

This work was supported under the National Science Council Grants 100-2218-E-011-008.

References

[1] P. Eckersley, "How unique is your web browser?" in *Privacy Enhancing Technologies*. Springer Berlin / Heidelberg, 2010, vol. 6205, pp. 1–18.

[2] 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*, ser. SIGMETRICS '98/PERFORMANCE '98. New York, NY, USA: ACM, 1998, pp. 11–21.

[3] S. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, mar 1999, pp. 227 –234 vol.1.

[4] T. Kohno, A. Broido, and K. Claffy, "Remote physical device fingerprinting," in *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, April-June 2005, pp. 93–108.

[5] S. J. Murdoch, "Hot or not: revealing hidden services by their clock skew," in *CCS '06: Proceedings of the 13th ACM Conference on Computer and Communications Security*, New York, NY, USA, 2006, pp. 27–36.

[6] S. Zander and S. J. Murdoch, "An improved clock-skew measurement technique for revealing hidden services," in *Proceedings of the 17th conference on Security symposium*. Berkeley, CA, USA: USENIX Association, 2008, pp. 211–225.

[7] D.-J. Huang, W.-C. Teng, C.-Y. Wang, H.-Y. Huang, and J. M. Hellerstein, "Clock skew based node identification in wireless sensor networks," in *IEEE Global Communications Conference*, 2008, pp. 1877–1881.

[8] M. Uddin and C. Castelluccia, "Toward clock skew based wireless sensor node services," in *Wireless Internet Conference (WICON), 2010 The 5th Annual ICST*, March 2010, pp. 1–9.

[9] Top threats to cloud computing v1.0. [Online]. Available: https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf

[10] J. Hall, M. Barbeau, and E. Kranakis, "Detection of transient in radio frequency fingerprinting using signal phase," in *Proceedings of IASTED International Conference on Wireless and Optical Communications (WOC '03)*, 2003.

[11] R. M. Gerdes, T. E. Daniels, M. Mina, and S. F. Russell, "Device identification via analog signal fingerprinting: A matched filter approach," in *Proceedings of the 2006 Network and Distributed System Security Symposium (NDSS '06)*, February 2006.

[12] K. Bonne Rasmussen and S. Capkun, "Implications of radio fingerprinting on the security of sensor networks," Sept. 2007, pp. 331–340.

[13] S. Jana and S. K. Kasera, "On fast and accurate detection of unauthorized wireless access points using clock skews," in *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking*, 2008, pp. 104–115.