On the Reliability of Wireless Fingerprinting using Clock Skews [*] [†]

Chrisil Arackaparambil, Sergey Bratus, Anna Shubina, and David Kotz
Dept. of Computer Science, Dartmouth College, Hanover, NH 03755, USA
{cja, sergey, ashubina, kotz}@cs.dartmouth.edu

ABSTRACT

Determining whether a client station should trust an access point is a known problem in wireless security. Traditional approaches to solving this problem resort to cryptography. But cryptographic exchange protocols are complex and therefore induce potential vulnerabilities in themselves. We show that measurement of clock skews of access points in an 802.11 network can be useful in this regard, since it provides fingerprints of the devices. Such fingerprints can be used to establish the first point of trust for client stations wishing to connect to an access point. Fingerprinting can also be used in the detection of fake access points.

We demonstrate deficiencies of previously studied methods that measure clock skews in 802.11 networks by means of an attack that spoofs clock skews. We then provide means to overcome those deficiencies, thereby improving the reliability of fingerprinting. Finally, we show how to perform the clock-skew arithmetic that enables network providers to publish clock skews of their access points for use by clients.

Categories and Subject Descriptors: C.2.1 [Computer-Communication Networks]:Network Architecture and Design

General Terms: Experimentation, Measurement, Security

1. INTRODUCTION

Clock skews are the inherent tiny drifts in the clocks of hardware devices due to variations in the manufacturing process. The use of clock skews of devices on a network for the purpose of fingerprinting those devices was first studied by Kohno, Broido, and Claffy [11]. They showed that it was possible to remotely measure the microscopic skews of devices, and that their fingerprinting method could identify individual devices despite errors inherent in remote measurements. Such fingerprinting has innumerable applications, e.g., in network forensics for identification, and also in penetration testing to identify network systems to know their weaknesses (the method of Kohno et al. can be used to identify virtual hosts served by the same physical device).

The study of Kohno et al. focused on the measurement of skews in wide-area networks by observing timestamps in TCP and ICMP packets. On the wireless side, Jana and Kasera [10] studied the approach of Kohno et al. at the MAC layer of 802.11 networks. They observed that, due to the essentially zero latency and the availability of a high frequency stream of precise beacon timestamps, the process of measuring clock skews became more accurate in these networks. They also showed that the clock skews of wireless devices remain consistent over time and changing external factors like temperature, and that skews vary across devices. The main application considered in their work was that of detecting fake APs. Today, tools like `rglueap` and `rfakeap` are readily available that make it easy for an attacker to set-up an AP that fakes a real one. Identifying fields in 802.11 frames like MAC address, BSSID, and SSID can be easily set to values desired by the attacker. A client attempting to associate with the real AP can be diverted to the fake AP, thus becoming vulnerable to various kinds of attacks. As pointed out before [9, 10], the attacker may also attempt to avoid detection of the fake AP by either operating on a channel different from the real AP, or by offering a stronger signal to the client.

Our contributions. In this work we show how previous methods for measuring clock skews are inadequate for fingerprinting and provide a means to overcome the problems that arise. Our work provides new insights into the implementation of the 802.11 standard in commodity hardware. In particular, we present

- a new method to measure clock skew, rather, a more precise clock to measure it against;
- an attack that spoofs the clock skew of a fake AP to mimic that of a real one, thereby rendering the two indistinguishable by the methods proposed previously;
- additional parameters to measure authenticity of the skew, enabling detection and mitigation of spoofing;

[*]An expanded version of this paper is available [1].

[†]This article results from a research program in the Institute for Security, Technology, and Society (ISTS), supported by the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

- clock skew arithmetic, that enables a network provider to publish skews of APs in the network independent of client stations.

The Role of Fingerprinting in Securing Wireless Infrastructure. Initially, 802.11 link layer security focussed on denying network access to unauthorized clients. The entire concept of 802.11 authentication, association, and in particular the design of the client state machine, proceeded from the assumption that the primary goal of the security mechanisms was to protect the network from rogue clients. The APs were thought of as the "perimeter" of the network, vested with the role of protecting it against rogue clients.

However, subsequent experience showed that the threat model underlying this design was inherently flawed. Clients (with stored representations of trust relationships) turned to be a much more important piece of the holistic security puzzle than previously thought. In fact, they emerged as the weakest link in the so-called perimeter. Exploiting network clients by tricking them into establishing connections to rogue services has become a leading strategy for both exploitation and penetration testing as evidenced by an entire BlackHat 2009 track (e.g., [13]) devoted to client exploitation functionality in the popular Metasploit penetration testing tool. It did not take long till the same attack approach was realized in 802.11 Layer 2. Crafting malformed inputs in frame fields quickly emerged as an extremely efficient attack methodology [6]. This methodology yielded such achievements as "hijacking a Macbook in 60 seconds" [5] (by way of a crafted probe response leading to attack code execution within the ring zero driver kernel context) and the subsequent automation and refinement of this technique that revealed other 802.11 driver vulnerabilities— the so-called "Month of kernel bugs" (see, e.g., [4]).

For setting up fake APs, popular exploitation tools, such as Karma [15], were developed to meet penetration testers' demand. Such early attacks were described by wireless security researchers in [9, 14]. However, such traditional fake AP scenarios assume successful establishment and maintenance of a layer 3 connection, whereas a new class of attacks is based on compromising the client at a much earlier point: either during scanning for available networks or during authentication or association attempts. As such, strong cryptographic schemes for authenticating access points, such as WPA2-Enterprise, cannot mitigate this threat. *Fake access points thus become a tool of delivering link layer exploits.*

As we have seen, establishing trust for an AP can be a tricky issue for a client. Traditional approaches to such trust-relationship problems most often find solutions in cryptographic exchange protocols. With respect to wireless security, the 802.11i RSNA (Robust Security Network Association) provides such a functionality. Importantly however, such protocols that are dependent on cryptography are complex and therefore induce potential vulnerabilities in themselves. These protocols must be implemented with great care. Before involving in complex cryptographic exchange protocols with an untrusted entity, we propose using clock-skew fingerprinting as a means of providing a first point of trust for clients. As such, we propose our methods as a complement to the existing authentication methods.

The following sections describe our contributions in detail.

2. MEASUREMENT OF CLOCK SKEWS

We first give an overview of the timing and synchronization processes in wireless networks as specified in the IEEE 802.11 standard. These processes provide the timing information required to compute the clock skews of APs.

In an 802.11 network operating in infrastructure mode, every station maintains a timer. This timer is synchronized with the timer in the AP the station is associated with via a Timer Synchronization Function (TSF). The synchronization is achieved through the beacon frames transmitted by the AP at periodic intervals. The most common setting for the beacon interval is 100 milliseconds. The beacon frames contain the TSF timer timestamp of the AP "at the time that the data symbol of the first bit of the timestamp is transmitted to the wireless medium," adjusting for hardware transmission delays. The timer is of microsecond resolution and is maintained as a 64-bit counter. Client stations set their local TSF timers to the values observed from beacon frames, again, adjusting for hardware delays. This means that beacon timestamps provide a high-precision mechanism to measure the skew in an APs TSF timer.

Clock skews. We now define the notion of clock skew as given by Moon, Skelly, and Towsley [12], and later used by Kohno et al. [11] and Jana and Kasera [10]. To measure the clock skew of an AP, we passively monitor the wireless interface of the measuring device for beacon frames from the AP. For beacon frame i we record the time t_i when it was received and the timestamp T_i in the beacon frame. In this manner we obtain a set of n measurements $(t_i, T_i), 1 \leq i \leq n$. We found that sampling $n = 100$ beacons gave sufficient accuracy in our experiments (as also observed previously [10]). We denote by x_i the elapsed time since the first beacon was observed, i.e., $x_i = t_i - t_1$. Similarly, let $w_i = T_i - T_1$. The quantity $y_i = w_i - x_i$ is called the clock offset of the ith measurement. In this way we get a set of n clock offset points (x_i, y_i). Ideally, there should be no relative skew between the measurer's clock and the beacon timestamps representing the AP's clock, when we would have $w_i = x_i, \forall\, i \leq n$. In reality we observe that the clock offset points lie on an approximately linear pattern that has some non-zero slope. A linear least square fit (LSF) is used to fit a line $y = s \cdot x + c$ to the set of clock offset points (y_i, x_i) by minimizing the least square error $\sum_{i=1}^{n}(y_i - (s \cdot x_i + c))^2$. The slope of the line obtained by LSF gives the clock skew of the AP. Skews observed in practice are tiny, but consistent, and are reported in parts per million (ppm).

Monitor mode synchronization. The timestamps in the beacon frames form one half of the required information for estimating clock skews. The measurement of the arrival time of the beacon frame is an important problem, since its accuracy impacts the accuracy of estimation. There were several clocks considered in [10] to report the beacon arrival time. The timestamp reported in the Radiotap header in the `pcap` field `radiotap.mactime` was considered. This timestamp is reported by the driver from the TSF timer maintained by the wireless hardware. But the approach was abandoned since the timer values were updated from the incoming beacon timestamps and hence did not serve as a stable clock. The approach finally found to work was to use the time reported by the Linux kernel via the function `do_gettimeofday()`.

However, this method too suffers from some drawbacks. The `do_gettimeofday` function is implemented using timer interrupts, and is adjusted in the kernel for anticipated delays. So it could be expected to shift in accuracy, and is only as accurate as the underlying interrupt mechanism. Also, since the skew of the clock represented by the function depends on the implementation of the function and underlying routines, we expect this skew to vary with the updates to the system. This would require clients to recalibrate their skew measurements before fingerprinting APs again. We have observed significant changes in the implementation of the `do_gettimeofday` function between kernel releases.

We present a new method of measuring the arrival time of beacons that is more accurate than using `do_gettimeofday`. Since this measurement is critical to estimating the clock skew, our method leads to more accurate measurements. Further the clock used in our method is implemented in the wireless hardware, and hence its skew does not change with software updates. Our method depends on the synchronization behavior of the Atheros chipset based cards that we use. We explain this behavior now. In the course of our discussion we state some observations and verify them empirically. These observations turn out to be crucial to our techniques described in later sections.

For the experiments in the rest of the paper we use two laptops as measurement stations. These laptops run the Ubuntu 9.04 GNU/Linux distribution (with kernel 2.6.28-15) and are each equipped with a wireless card based on the Atheros 5212 chipset. The Madwifi driver `ath_pci` is used with these cards for the measurements. In our experiments we also use two Linksys APs, henceforth referred to as Linksys 1 and Linksys 2 respectively.

The monitoring performed to capture the beacon frames is done in the monitor-mode of the wireless interface. The TSF timer maintained in the wireless hardware is a high-accuracy microsecond resolution timer, and it would serve best for our measurements of beacon arrival time, since its value is provided directly to the driver by the hardware and is not affected by other processes in the system. However, this timer was deemed as unusable in [10] because the timer was kept synchronized to the incoming beacon timestamps even in monitor mode. We now give a method to use this timer. It should be noted that in monitor mode, it is not necessary to synchronize the TSF timer with the incoming beacon timestamps, since the card is completely passive in this mode. However, cards with the Atheros chipset continue to synchronize with the beacon frames observed from the AP that the card was *last* associated with. This leads to an interesting possibility: what happens when the AP that the card was last associated with becomes inactive and stops broadcasting beacon frames? In this case the timer on the card, not being able to synchronize with the beacon timestamps, should begin to drift with its *own* skew. And indeed, this is confirmed empirically with our experiments. We measured the skew of Linksys 1 and Linksys 2 in monitor mode, by first associating the measuring laptop with Linksys 2 and then switching the laptop to monitor mode. We then turned Linksys 2 off and again measured the skew of Linksys 1. Figure 1 shows the clock offsets points measured in the different cases, and Table 1 reports the measured clock skews. Observe that the estimated skew of Linksys 1 varied significantly before and after Linksys 2 was turned off. Also, the estimated skew of Linksys 2 was negligible. Note that to similarly measure the skew of Linksys 2 when the associated AP was turned off, we had to first associate with Linksys 1 and repeat the process. This leads us to the following observations.

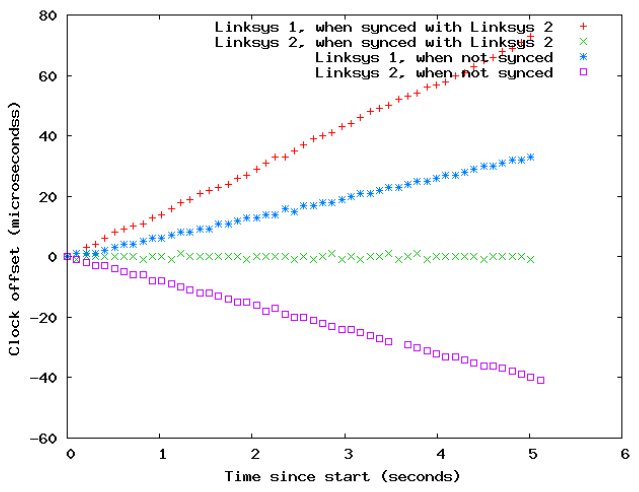

Figure 1: Clock offsets with and without the measuring station syncing its TSF timer with Linksys 2.

	Clock Skew
Linksys 1, when synced with Linksys 2	14.37
Linksys 2, when synced with Linksys 2	-0.01
Linksys 1, when not synced	6.68
Linksys 2, when not synced	-7.85

Table 1: Clock skew measurements with sample sizes of 100 beacon timestamps, with and without synchronization with Linksys 2.

OBSERVATION 1. *Given a wireless card in Station mode and associated with an AP A, when the card is switched to Monitor mode, it continues to update its TSF timer register with the beacon timestamps from AP A.*

OBSERVATION 2. *Given a wireless card in Station mode and associated with an AP A, when the card is switched to Monitor mode, if AP A ceases to transmit beacons, then the TSF timer maintained in the wireless card begins to drift with its own, actual skew.*

The next two observations follow from Observation 1.

OBSERVATION 3. *Given a wireless card in Station mode and associated with an AP A, when the card is switched to Monitor mode, the clock skew of AP A as measured by the card is zero (imperceptible).*

Observation 3 suggests that the skew of the measuring card becomes equal to the skew of the AP it is synchronized with. From Table 1 it may further be observed that the skew of Linksys 1 when measured by the card synchronized with Linksys 2 is approximately equal to the difference of the skews of Linksys 1 and Linksys 2 when there is no synchronization. We have observed this behavior consistently with different APs; we omit the data for the sake of brevity. This indicates that it is possible to compute the skew of a wireless device as measured by another, by passively measuring the skews of the two devices.

Observation 4. *Given a wireless card in Station mode and associated with an AP A, when the card is switched to Monitor mode, the clock skew of another AP B as measured by the card is equal the skew of AP B as would be measured by AP A.*

The issue of performing arithmetic to determine the skew between a pair of wireless devices is detailed in Section 5.

Our measurement technique. The previous observations give us a new method of measuring beacon arrival times—using the TSF timer to do it. For the experiments in the rest of the paper that use the TSF timer, we use the timer by first associating with an AP and then switching off power to the AP. On a client station the same effect can be achieved by either removing and re-inserting the wireless card, or even through software by reloading the driver modules. It may even be possible to power-cycle the card and flush the state through the driver interface, but we have not verified this. Our next experiments show that using the TSF timer yields much higher accuracy than using `do_gettimeofday`. To compare the two methods we collected 10 sets of beacon traces with each method. As in [10] we disable NTP to avoid its effect on the `do_gettimeofday` method. For each set of traces we measured the clock skew of Linksys 1 using sample sizes ranging from 100 beacons to 600 beacons. Then for each set of traces, and each sample size we computed the mean and variance of the clock skew. Our observations are presented in Table 2 and Figure 2 Observe that the variance of the clock skew when using the TSF timer is consistently several orders of magnitude smaller that the variance when using `do_gettimeofday` function to report the beacon arrival times. This points to the superior stability and accuracy of the TSF timer method.

	Mean	Variance
TSF Timer	6.7011	0.0001245
`do_gettimeofday`	-28.1347	0.0659

Table 2: Mean and variance of 10 clock skew measurements (ppm) with the two clocks. Beacon timestamp sample size is 100.

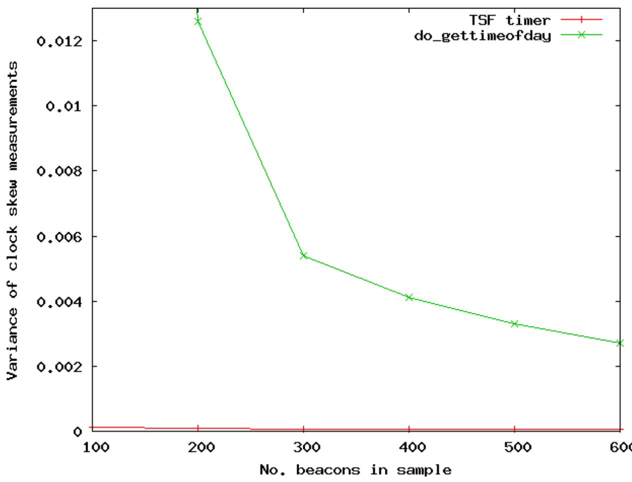

Figure 2: Variance in clock skew measurements as a function of the beacon timestamp sample size.

3. VULNERABILITY OF PREVIOUS MEASUREMENT METHODS

In this section we present a spoofing attack that is able to fool the method of [10] that relied only on the clock skew measurement to detect spoofing. Our technique finds its basis in two key points:

1. Observations 3 and 4 show that a measurement device measures different clock skews depending on whether its TSF timer is synchronized with the beacon timestamps from an AP, because that timer, being synchronized, *acquires* the skew of the AP.

2. The Madwifi driver allows the multiple creation of virtual interfaces (VAPs) for a single physical device. These virtual interfaces may be in different modes—station, master, or monitor—and in particular, one station VAP is allowed to exist along with several AP VAPs. These virtual interfaces can then be brought "up" to begin operation.

These points suggest that we might be able to have an AP VAP and a station VAP, with the station VAP associated with the real AP, and the AP VAP configured as the fake AP. Since the two interfaces would share the same hardware TSF timer, the timer would acquire the skew of the real AP due to the station VAP associating with it. This skew would be reflected in the timestamps in the beacons emitted by the AP VAP, thereby spoofing the clock skew of the real AP. However, carrying out the above attack required modification of the Madwifi driver. Details of the modification are in the full version of the paper [1]. Table 3 shows the skews from four measurements. It can be seen that it is not possible to detect the fake AP by comparing the clock skew alone, with a reasonable degree of certainty. Figure 3 shows the clock offset points from the two APs. The synchronization behavior produces periodic dips in the plot. In Section 4 we show how to capture this behavior to measure of the reliability of the clock skew. The authors of [10] present several arguments showing why the skews of APs cannot be fabricated. The failing assumption made in their arguments is that the attacker, on knowing the clock skew of the the real AP, would need to perform arithmetic with his local timer values to compensate for his own skew. Our attack does not measure the skew of the real AP in advance and try to compensate for it, so we do not suffer from the effects of transmission delays and computational overheads.

Real skew	Real intercept	Fake skew	Fake intercept
16.79	0.53	16.78	-2.13
16.82	0.51	16.69	1.43
16.80	-0.02	16.74	-1.34
16.81	0.17	16.78	-1.10

Table 3: Skews and intercepts from real & fake AP.

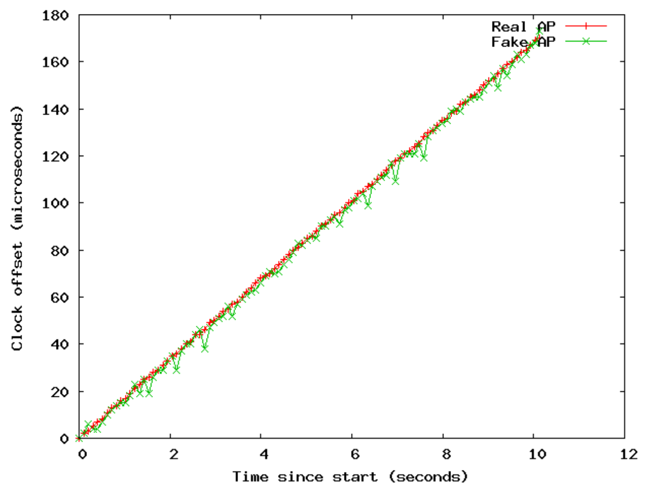

Figure 3: Clock offsets with 100 beacon timestamps from the real AP and the fake AP.

Real skew	Real intercept	Fake skew	Fake intercept
17.35	-0.78	16.29	71.49
17.29	0.70	17.58	-10.63
17.26	-0.46	17.54	-10.49
17.25	-0.19	16.49	-19.53

Table 4: Clock skews and line intercepts from the real and *bridged* fake AP.

Extending the scope of the attack

We now show how the scope of the attack can be extended by using a "bridge" AP. The function of the bridge AP is to allow the attacker to move the fake AP to cover a wider range (perhaps in order to be out of range of the real AP), or to operate in a different channel, all while still spoofing its clock skew. The bridge AP synchronizes its TSF timer with that of the real AP as described earlier, and the fake AP synchronizes its timer with that of the bridge AP. To operate on a different channel we take advantage of the fact that frequency ranges of adjacent channels as prescribed by the 802.11 standards overlap.

In our experiment we have the real AP operating on channel 11, the bridge AP on channel 10, and the fake AP on channel 9. Our results from four traces are shown in Table 4 and Figure 4. As it may be expected, the quality of spoofing degrades due to the bridging, but the clock skew of the fake AP is still fairly close to that of the real AP.

4. IMPROVING THE RELIABILITY OF FINGERPRINTING

We now present techniques to mitigate the risks of attacks like those presented in the last section, by gauging the reliability of the measured clock skews.

Line-fitting error. The most straightforward approach is to measure the error in line fitting. We observed that the spoofing attack in the previous section introduced an artifact—the dips in the plots of clock offset points in Figures 3 and 4. There are several ways to measure these fluctuations. First, we consider the y-intercept c of the fitted line $y = s \cdot x + c$.

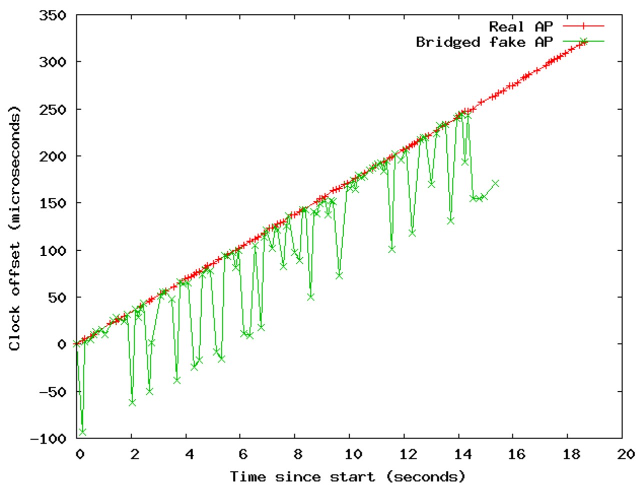

Figure 4: Clock offsets with 100 beacon timestamps from the real AP and the *bridged* fake AP.

Since, in the ideal case, the line passes through the origin, the absolute value of the intercept serves as one parameter to measure the fitting error. Tables 3–4 show the values of the parameter c with our attacks. The absolute value of c for the fake AP is higher than that for the real AP.

We also consider the jitter of the beacon timestamps as a means to measure clock skew reliability. Given a set of clock offset points, the jitter is $\gamma = \sum_{i=1}^{n-1} |y_{i+1} - y_i|/(n-1)$ and provides a measure of the temporal variations in the beacon timestamps. We defer the measurements of jitter in our attacks to the next section, where we analyze the effect of beacon interval on our attacks.

Beacon interval	c real	c fake	change	γ real	γ fake	change
25	0.20	2.13	965%	0.50	3.1	520%
50	0.58	1.23	112%	0.84	3.34	298%
100	0.31	1.50	384%	1.71	3.11	82%
200	0.40	1.68	320%	3.43	3.88	13%

Table 5: Variation in parameters c and γ with different values of the beacon interval.

Analysis of beacon-interval on skew measurements. The value of the beacon interval of the AP affects the ability of the attacker to spoof its clock skew with our attack. When the beacon interval is set to smaller values, the attacker needs to present a finer-grained clock via the beacon timestamps. At lower beacon intervals the fluctuations in the synchronized clock of the attacker become more prominent since the various processing delays play a relatively larger role. To validate this hypothesis we perform our attack with different settings of the beacon interval parameter, and measure the parameters c and γ described earlier for testing the reliability of clock skew measurements. On plotting the clock offset points for this experiment (see full version [1]), we observe that the dips in the plots (such as those seen in Figure 4) increase in magnitude as the beacon interval is reduced. In Table 5 and Figure 5 we show the variation of the parameters c and γ with different beacon intervals. We observe

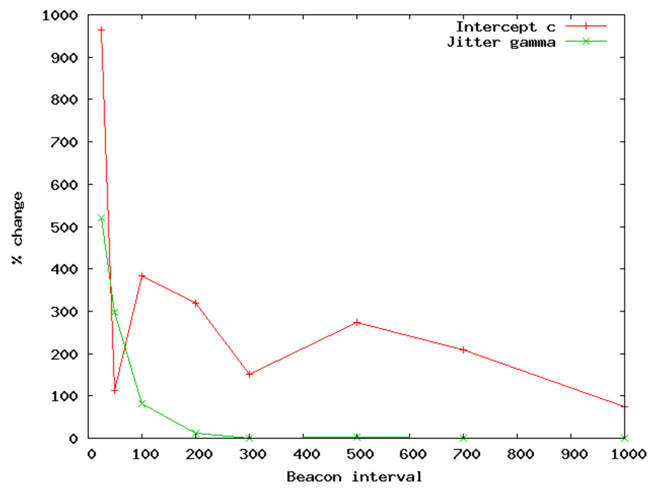

Figure 5: Variation in parameters c and γ with different values of the beacon interval.

that the changing magnitude of the dips is captured very well by the jitter parameter γ, and to a good extent also by the intercept c. Thus, to avoid clock skew spoofing attempts it is advisable to use a beacon interval that is as small as permissible, and use the parameters c and γ to gauge the reliability of clock skew measurements.

5. SKEW ARITHMETIC

In this section we show how to perform arithmetic with clock skews. For example, if we know the clock skew s_{AB} of AP B as would be measured by AP A, then we can compute the skew s_{BA}, i.e., the clock skew of AP A as would be measured by AP B. After doing the necessary derivation and approximation (see full version [1] for details), we find that the intuitive formula $s_{BA} = -s_{AB}$ turns out to work well in theory and in practice. We can also compute the clock skew s_{BC} of AP C as measured by AP B, when given the clock skews s_{AB} and s_{AC}. After doing necessary derivation and approximation [1] we again find that that the intuitive formula $s_{BC} = s_{AC} - s_{AB}$ works well in theory and in practice.

Unlike Jana and Kasera's proposal [10], where the fake AP detection procedure was meant to be implemented in a Wireless Intrusion Detection System (WIDS) node, in our work we even enable the measurements to be done on wireless clients themselves. This requires the use of skew arithmetic. To determine whether to trust an AP by measuring its clock skew, the client must know the skew of the real AP beforehand. Since the clock used by the client has a skew of its own, it would be necessary for the client to have measured the skew of the real AP using its own clock beforehand. However, the ability to perform skew arithmetic eliminates this requirement. Network providers can publish the skews of the APs in their network as measured against a high-precision clock of negligible skew. Then to measure the skew of an AP using skew arithmetic, the client only needs to know the skew of its own clock against a similar high-precision clock. The process is simplified further if network card vendors measure and publish the skews of the cards they produce at the time of testing.

6. RELATED WORK

Existing methods of *passive* L2 fingerprinting of 802.11 client stations aimed to improve client identification for defensive or forensic purposes by verifying facts about the client. Franklin et al. [8] fingerprinted clients based on the clients' driver-specific probing behavior, and Ellch [7] fingerprinted clients based solely on statistical distributions of the 2-byte NAV field in established client connections.

A passive method for detecting fake APs was presented by Bahl et al. [2]. The anomaly in sequence numbers in beacons from the real and fake APs was used to detect the fake AP. For the case when the two APs are not active together, location-based detection was suggested. Still, it was observed [10] that even such methods are not very reliable, and fail if the attacker is able to position his AP carefully.

Bratus et al. stress [3] the importance of *protecting clients from APs* in the early stages of connection before cryptography based trust in the AP could be established, and proposed an *active* fingerprinting scheme that tested certain properties of the AP before accepting complex data from it.

7. REFERENCES

[1] Chrisil Arackaparambil, Sergey Bratus, Anna Shubina, and David Kotz. On the Reliability of Wireless Fingerprinting using Clock Skews. Technical Report TR2010-661, Dartmouth College, Computer Science, Hanover, NH, January 2010.

[2] Paramvir Bahl, Ranveer Chandra, Jitendra Padhye, Lenin Ravindranath, Manpreet Singh, Alec Wolman, and Brian Zill. Enhancing the security of corporate wi-fi networks using dair. In *MobiSys '06: Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 1–14, New York, NY, USA, 2006. ACM.

[3] Sergey Bratus, Cory Cornelius, David Kotz, and Daniel Peebles. Active behavioral fingerprinting of wireless devices. In *WiSec '08: Proceedings of the first ACM conference on Wireless network security*, pages 56–61. ACM, 2008.

[4] Laurent Butti. Wi-Fi advanced fuzzing. Black Hat Europe, February 2007.

[5] Johnny Cache and David Maynor. Hijacking a MacBook in 60 seconds. Black Hat, August 2006.

[6] Johnny Cache, H D Moore, and skape. Exploiting 802.11 wireless driver vulnerabilities on Windows. *Uninformed.org*, 6, January 2007.

[7] J.P. Ellch. Fingerprinting 802.11 devices. Master's thesis, U.S. Naval Postgraduate School, September 2006.

[8] Jason Franklin, Damon McCoy, Parisa Tabriz, Vicentiu Neagoe, Jamie Van Randwyk, and Douglas Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. In *Proceedings of 15th USENIX Security Symposium*, pages 167–178. USENIX, August 2006.

[9] The Shmoo group. 802.11 bait, the tackle for wireless phishing. Toorcon, October 2005.

[10] Suman Jana and Sneha Kumar 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*, pages 104–115. ACM, 2008.

[11] Tadayoshi Kohno, Andre Broido, and K. C. Claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2):93–108, 2005.

[12] S.B. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. In *IEEE INFOCOM '99*, volume 1, pages 227–234, March 1999.

[13] H D Moore. Mastering the Metasploit framework. http://www.blackhat.com/html/bh-usa-09/train-bh-usa-09-hdm-meta.html, July 2009.

[14] Simple Nomad. Hacking the friendly skies. Shmoocon, January 2006.

[15] Dino A. Dai Zovi and Shane "K2" Macaulay. Karma. http://trailofbits.wordpress.com/karma/.