Chapter 20

DETECTING NON-DISCOVERABLE BLUETOOTH DEVICES

Daniel Cross, Justin Hoeckle, Michael Lavine, Jason Rubin and Kevin Snow

Abstract Mobile communication technologies such as Bluetooth are becoming ubiquitous, but they must provide satisfactory levels of security and privacy. Concerns about Bluetooth device security have led the specification of the "non-discoverable" mode, which prevents devices from being listed during a Bluetooth device search process. However, a non-discoverable Bluetooth device is visible to devices that know its address or can discover its address. This paper discusses the detection of non-discoverable Bluetooth devices using an enhanced brute force search attack. Our results indicate that the average time to attack a non-discoverable Bluetooth device using multiple search devices and condensed packet timing can be reduced to well under 24 hours.

Keywords: Bluetooth security, device discovery, non-discoverable mode

1. Introduction

Bluetooth devices are growing in popularity, despite security concerns. A Bluetooth device in the "discoverable" mode is easily scanned using a computer; moreover, private information can be downloaded from the device. This technique has been used in high profile attacks against celebrities whose devices were operating in the discoverable mode.

To address some of these concerns, the Bluetooth Special Interest Group (SIG) recommends that devices be placed in the non-discoverable mode, which prevents them from being listed during a Bluetooth device search process. However, a non-discoverable Bluetooth device can still be attacked if its address is already known or is determined by brute force. Most brute force methods take about a week, which alleviates the security concerns to some extent. However, a brute force search for non-discoverable Bluetooth devices can be sped up significantly using certain details about device operation.

This paper presents a novel technique for detecting non-discoverable Bluetooth devices. It leverages the constraints imposed on the connection process and uses multiple search devices to enhance the brute force search for device addresses. Results indicate that the average time to successfully attack a non-discoverable Bluetooth device using 79 search devices and condensed packet timing is well under 24 hours.

2. Background

The Bluetooth specification [1] establishes certain constraints and parameters to ensure the interoperability of Bluetooth devices. Our method for accelerating the search for non-discoverable devices leverages the constraints on the Bluetooth connection process.

2.1 Bluetooth State Transitions

The Bluetooth specification defines a state transition sequence for Bluetooth devices. A device in the Standby state can enter the Inquiry, Inquiry Scan, Page or Page Scan states. A Bluetooth device enters the Inquiry state when it sends inquiry packets. A device enters the Inquiry Scan state to listen for and respond to inquiry messages broadcast by other devices. The Inquiry and Inquiry Scan states are used for device discovery. The Page and Page Scan states are reached when a connection is being established between two devices. A Bluetooth device enters the Page state when it transmits page packets; it enters the Page Scan state to listen for and respond to page packets with its address. The Master Response and Slave Response states are entered when packets are exchanged as a connection is being established. Upon successfully completing a connection, a Bluetooth device enters the Connection state.

2.2 MAC Address Components

Every Bluetooth device is assigned a unique 48-bit MAC address. The 48-bit address has three segments. The sixteen most significant bits are dedicated to the non-significant address part (NAP). The upper address part (UAP) constitutes the next eight bits of the address. The NAP and UAP are assigned to companies that manufacture Bluetooth devices. The last 24 bits of the address make up the lower address part (LAP), which is designated by the manufacturer. The addressing of page packets is based on the 24-bit LAP.

2.3 Discoverable and Non-Discoverable Modes

Bluetooth devices discover and connect to each other using inquiry and paging procedures, respectively. An inquiry procedure is initiated by an inquiring device. Discoverable Bluetooth devices send responses to inquiry packets, making the inquiring device aware of its presence. Bluetooth devices in the non-discoverable mode do not reply to inquiry packets and, thus, remain invisible to the inquiring device.

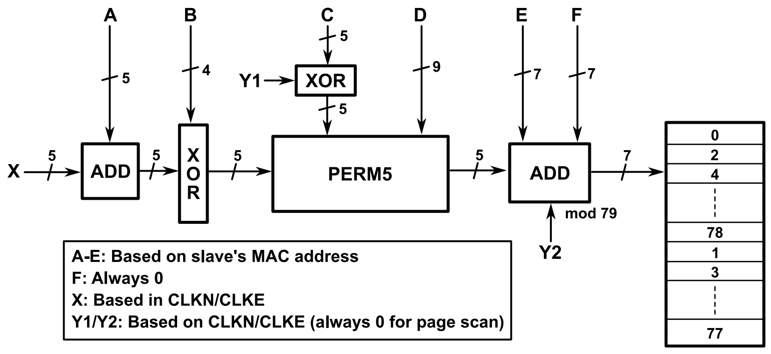

Figure 1. Hop selection kernel for page, page scan and response.

2.4 Connection Process

The paging procedure uses the address of a nearby Bluetooth device. Two devices are involved in a paging procedure: the device that seeks a connection by issuing a paging packet and the device that listens for a paging packet using a page scan. Only the device with the correct address responds to a paging packet. The channel used by the paging procedure is dependent on the characteristics of the connectable device.

The paging device attempts to approximate the clock of the connectable device by estimating its offset and adding the offset to its own clock. This approximation determines the timing of the page scan channel of the connectable device. The hopping sequence is determined by the address of the connectable device. The page hopping sequence and page response hopping sequence both utilize only 32 of the 79 available frequencies. The frequencies used by the two sequences are in one-to-one correspondence with each other.

2.5 Hop Channel Calculation

A Bluetooth device calculates a hop channel on-the-fly as it is dependent on the time and the address of the device with which it is communicating. The set of channels and the order in which they are used are determined by the hop selection kernel (Figure 1). According to the Bluetooth specification, the order and phase are determined by portions of the lower 28 bits of the slave's MAC address and the paging device's clock. The lower seven odd bits of the MAC address are used to determine the channels in the set. The inputs X, Y1 and Y2 vary depending on the current mode (page scan, inquiry scan, page, inquiry, responses, etc.). Inputs A–F remain constant for the majority of modes.

2.6 Page Packets

A page packet is 68 bits in length. Unlike other packets, it has no header, payload or trailer, only a device access code (DAC). The DAC consists of a 4-

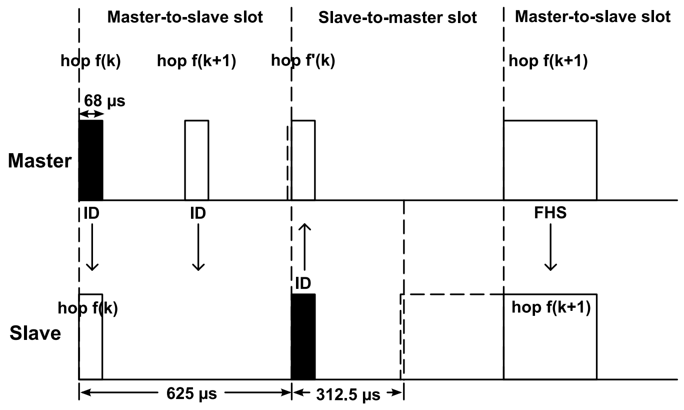

Figure 2. Paging process (first packet).

bit preamble and a 64-bit sync word. The preamble is an alternating sequence of `0`s and `1`s based on the least significant bit of the sync word. The sync word is created by appending six bits to the end of the LAP. If the most significant bit of the LAP is `0`, then `001101` is appended to the LAC; otherwise, `110010` is appended. The thirty bits are then XORed with a pseudo-random noise (PN) sequence. The result is then XORed with a previously generated codeword to create the 64-bit sync word.

2.7 Paging Process

The paging process hops channels 3,200 times per second. Two page packets are sent in every transmission (TX) slot, each on a different channel. A single TX slot covers 625 μs. Since there are two packets, 312.5 μs are used for each frequency. Out of the 312.5 μs, 68 μs are spent sending a packet.

Following every TX slot is a reception (RX) slot, which also lasts 625 μs. The RX slot is used by a paging device to receive the first page slave response packet from a connectable device. The paging response procedure used by the connectable device hops the same channels as the paging procedure and at the same rate. Upon receiving the page packet in the RX slot, the connectable device waits until it receives the corresponding packet in the next TX slot. If the connectable device receives the page packet in the second 312.5 μs half of its RX slot, then the first page slave response packet is sent in the second 312.5 μs half of the next TX slot. Thus, there is a consistent 625 μs between when the page packet is sent and when the first page slave response is received by the paging device.

Figures 2 and 3 illustrate the initial packet exchange in the paging process. The diagrams show that a reply to a page packet is sent 625 μs after the page packet. As shown in Figure 2, a page packet transmitted by a paging device as the first page packet in the TX slot results in a page reply 625 μs later in the

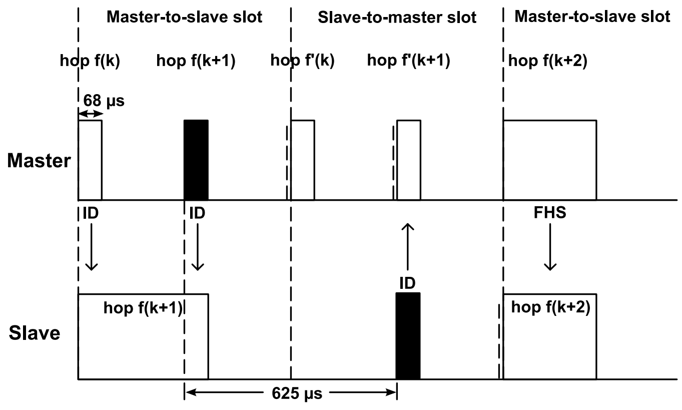

Figure 3. Paging process (second packet).

paging device's RX slot (also the first TX slot of the paged device). Similarly, a page packet transmitted as the second page packet in the TX slot results in a page reply 625 μs later in the paging device's RX slot (Figure 3). The paging process involves additional packet exchanges, but they are not relevant because our goal is to discover devices, not to establish connections with them.

3. Related Work

While a variety of *ad hoc* tools have been developed for brute force searches of non-discoverable Bluetooth devices, little research exists on enhancing brute force techniques. One exception is the work of Haataja [4]. Haataja reasons that the address space to be searched can be reduced by assuming that the manufacturer is known; this assumption yields a reduced search space of 2^{24} addresses. Haataja used a Bluetooth-compatible radio unit and a protocol analyzer to attempt ACL link connections to remote devices for each address. In one experiment, 2,000 addresses were scanned in 174 minutes. Based on these results, it is estimated that, on the average, 1.4 years of scanning would be required to discover a single device. Using 25 scanning devices would reduce the average time to 20.3 days.

However, some of Haataja's assumptions are problematic. First, it is not reasonable to assume that the device manufacturer is known. It should be possible to detect devices from any manufacturer. Therefore, the approach requires at least 20.3 days for each manufacturer. Our methodology, on the other hand, is faster and makes no assumptions about the device manufacturer.

The second problem is that the 20.3 day estimate relies on 25 devices scanning in parallel. Bluetooth operates on 79 unique frequencies and a reliable brute force search of one address requires connection attempts over 32 unique frequencies within a short time period. Parallel scanning offers no guarantees that the devices will not concurrently send requests on identical frequencies, which results in collisions. Thus, using 25 or more scanning devices in parallel is very unreliable due to the large number of collisions. In contrast, our methodology can employ up to 79 parallel devices while guaranteeing that no collisions will occur.

The security of Bluetooth devices in the non-discoverable mode is a serious issue. Wong and Stajano [10] have investigated the paging process and the security impact on Bluetooth devices operating in the non-discoverable mode. Their work is particularly relevant because the non-discoverable mode is the principal defense against tracking, monitoring and exploiting Bluetooth devices. Wong and Stajano also propose an enhanced anonymity method for connecting to devices using paging packets. Their Protected Pseudonyms method employing authentication and cryptography is one of a few proposals that enhances the security of the paging process and the non-discoverable mode.

A white paper by Gehrmann [3] focuses on the encryption and authentication architectures built in the baseband layer of Bluetooth devices. The paper discusses the need to keep a Bluetooth device in a secure environment when pairing. But it does not discuss the use of the non-discoverable mode as a security function, nor does it provide recommendations for using the mode to secure devices.

A more recent document released by the Bluetooth SIG [2] addresses several security exploits and concerns. The document explicitly cites the non-discoverable mode as the chief security mechanism and advocates its use to protect against a number of exploits and attacks against Bluetooth-enabled devices. It even states that the non-discoverable mode can protect devices from viruses and worms. Unfortunately, this premise is based on the assumption that a non-discoverable Bluetooth device cannot be detected and will not respond to an attacking device. Our methodology, which is capable of detecting Bluetooth devices in the non-discoverable mode, shatters this assumption.

4. Detection Methodology

A straightforward brute force search of the address space based on the Bluetooth specification is trivial, but incredibly time consuming. The best brute force approach [4] is estimated to take an average of 1.4 years using one search device, and just under a week with 79 devices (the maximum number of devices used in our methodology). Our methodology is faster because it: (i) reduces the number of addresses to be searched, and (ii) decreases the time taken to search addresses. Searching one address entails sending the correct page packet on each page scan frequency with the appropriate timing. Therefore, a brute force technique involves not only searching for all possible addresses, but also each of their corresponding paging channels.

The following sections describe our address space and search time reduction techniques. The results obtained using the reduction techniques are presented along with some practical considerations.

4.1 Address Space Reduction

The number of addresses to be searched is determined by the DAC sent in a page packet and the address inputs into the channel hop selection kernel. As mentioned above, one component of the DAC is the 24-bit LAP, which is the only portion of the address included in a page packet. Since there are 2^{24} unique LAPs, only 2^{24} addresses instead of all 2^{48} addresses need to be searched. Unfortunately, this also means that the number of addresses to be searched is at least 2^{24} without detailed address space profiling. However, paging a single address involves transmitting the page packet at the correct time on the appropriate channel.

As mentioned in Section 2, 28 bits of the address are used for hop channel selection. Using a brute force technique to identify the address corresponding to a channel set increases the search space from 2^{24} to 2^{28}. However, this increase is avoided because only certain bits of an address are used to select the channels in the page scan channel set. Specifically, the lower seven odd address bits 13, 11, 9, 7, 5, 3 and 1 are used to determine the channel set. Some of the remaining address bits determine the order of the channels selected, but this is not relevant to our methodology.

Since the address bits mentioned above are contained in the LAP, the size of the search space is still 2^{24}. Assuming a uniform distribution of address assignments, an average of 2^{23} addresses need to be searched before a response is received.

4.2 Search Time Reduction

Search time reduction is achieved through two approaches: condensed packet timing, which increases the rate at which page packets are sent; and parallelized paging, which enables simultaneous page requests over each page channel in one TX slot.

Condensed Packet Timing To decrease the amount of time taken to search an address, the number of packets sent per page scan window (N_{pw}) can be increased from the value in the Bluetooth specification. N_{pw} is computed using the following equation:

$$N_{pw} = \left\lfloor \frac{T_{w\,page\,scan}}{T_{page} + T_{delay}} \right\rfloor .$$

Figure 4 presents the timing diagram represented by the equation. Since the page scan window ($T_{w\,page\,scan}$) and the paging packet transmit time (T_{page}) are variables outside the scope of a paging device's control, the delay between page packets (T_{delay}) is the only controllable factor for determining the number of packets that can be sent in a single page scan window. In a standard page sequence, packets are sent at a rate corresponding to a T_{delay} of 244.5 μs. This allows the sending device enough time to change channels if needed. However,

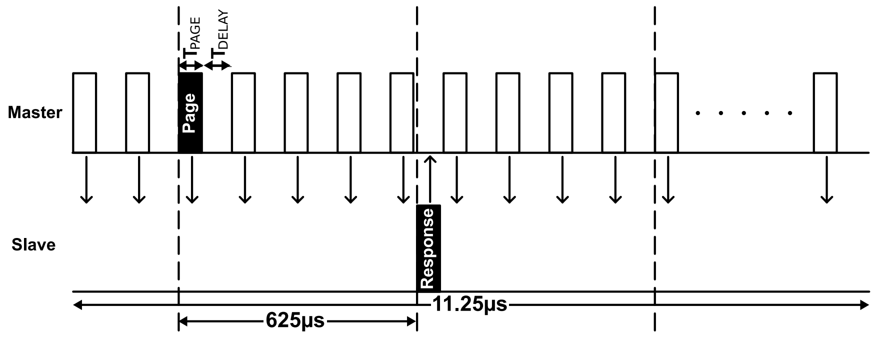

Figure 4. Page packet timing diagram.

our modified approach does not require a paging device to change channels; this permits the use of a lower T_{delay}.

Certain restrictions are imposed on T_{delay}. One is that T_{delay} must be large enough to allow a response packet to be received without colliding with other page packets. Since a page packet takes 68 μs to transmit, the lower bound on T_{delay} is 68 μs.

Another restriction is that the period during which a page response is returned (and no transmission is allowed) must be 625 μs after the respective page packet is sent to ensure that the packet is received. Therefore, a targeted device must not transmit during periods when a response page packet might be sent.

The minimum T_{delay} is 68 μs because a page packet must fit in the reception window. This yields 82 packets per page scan window. Unfortunately, transmitting and receiving packets during these intervals causes packet collisions. Since the number of packets must be reduced in increments of one, the number of packets per page scan window is reduced from 82 to 81, yielding a T_{delay} of 70.88 μs, which does not cause packet collisions.

Thus, a T_{delay} value of 70.88 μs is selected. It satisfies all the constraints and is significantly smaller than the value in the Bluetooth specification (244.5 μs). The result is that condensed timing allows more page packets to be sent in a single page scan window. Since we use the default page scan window value of 11.25 ms, N_{pw} is computed as:

$$\begin{aligned} N_{pw} &= \left\lfloor \frac{11.25\ ms}{68\ \mu s + 70.88\ \mu s} \right\rfloor \\ &= 81\ pages/window. \end{aligned}$$

Note that there is no way to control when a target device starts its page scan. This can occur while a packet is being transmitted, which would cause the device to miss a packet addressed to it. If the first and last packets sent in a single window are the same, then all unique packets (N_{upw}) could be received by the target device, but this comes at the expense of an additional packet per page scan window:

$$\begin{aligned} N_{upw} &= N_{pw} - 1 \\ &= 81\ pages/window - 1\ redundant\ page/window \\ &= 80\ pages/window. \end{aligned}$$

Parallelized Paging In addition to increasing the number of packets that are transmitted per page scan window via condensed timing, multiple page packets may be sent in parallel during a single TX slot. A device may be listening on any of the 79 possible page scan channels depending on its address, each of which can be transmitted simultaneously using multiple devices. Previous approaches have failed to address the fact that collisions increase as more devices are added to increase the scanning speed. Collisions decrease the reliability of address space scanning, which ultimately increases the search time.

These problems can be overcome using a simple matrix representation that models more complex timing and channel selection issues. Based on the assumption that 79 devices are being used and that each is transmitting in blocks of N_{upw}, matrices of size $N_{upw} \times 79$ can be created with <address, channel> pairs as their elements. Each column of a matrix is deemed to be a "transmission slot." The task is to fill the matrices as densely as possible. The next two sections describe how matrix elements are generated and how the elements are placed in matrices.

Paging Channel Set Generation A formula for generating the 32 paging channels for each address is a prerequisite for packet scheduling. We have previously described the main aspects of hop channel selection. Recall that it requires the set of channels, not the specific order of the set. Inputs E and F in Figure 1 are the only inputs that determine the unordered set of channels. Since input F is used in the Connection state, not in any of the paging sequence states, this leaves input E and the 5-bit output of a permutation to determine the channel set. A simple iteration from 0 to 31 (representing the 32 possible permutation values) added to E gives the 32 indices into a channel mapping register. Therefore, the formula for calculating the paging channel set given $i \in 0..31$ is:

$$Channel_i = (((E + i)\ mod\ 79) \times\ 2)\ mod\ 79.$$

$(E + i)\ mod\ 79$ generates the output of the final addition of the selection box. The modulo 79 computation maps the output of the addition (channel index) to the mapping register, producing the actual channel number.

A set of <address, channel> pairs is generated for each address. Since each 7-bit E value corresponds to a set of 32 channels, 128 groups with 32 channels each are created. Some of these sets are identical because of the modulo 79 computation. Each LAP address therefore belongs to one of the 128 groups. Thus, $2^{24}/128$ or 2^{17}addresses belong to each group, which must then be scheduled into matrices.

Packet Scheduling The goal is to fill all the matrices as densely as possible, where each matrix element corresponds to an <address, channel> pair. Two restrictions apply: (i) channels must be unique within a transmission slot (matrix column), and (ii) <address, channel> pairs for an address must fit in one matrix or be divided among matrices in multiples of 32.

The first restriction guarantees that no collisions occur between targeted devices. The second ensures that the listening device can successfully receive a page packet for each address. Since a device could be listening on any of its 32 paging channels during each page scan window (matrix window), each channel must be tried. However, the listening device increments the channel it listens on for each new matrix window. If half the channels for an address are tried in a matrix, the second half cannot be tried in a second matrix because nothing is known about the channel set order or phase. The only assumption that can be made is that 32 matrices later, the listening device will be on the same channel. This is because a listening device repeatedly scans the same 32 channel sequence.

A matrix can be created by scheduling the 32 channels of each of the 128 groups mentioned above into one matrix. Each element of the matrix contains the group number (0–127). This fills most of the first 64 of 80 slots, leaving the remaining 16 slots unused. The result is convenient because 16 is one quarter of 64: one of the 64-slot matrices can be split up into quarters, which can fill the remaining 16 slots of 4 sequential matrices. Thus, five 64-slot matrices fit into four 80-slot (full) matrices. The four full matrices may be used as a template by replacing the group number in each element with an <address, channel> pair from the respective group.

Unfortunately, this does satisfy the restriction that an address split across matrices must be 32 matrices apart. Therefore, instead of splitting one 64-slot matrix among four sequential full matrices, 32 64-slot matrices are split among 128 full matrices. The first quarters of the 32 64-slot matrices are split among the first 32 sequential full matrices, the 32 second quarters are split among the second 32 sequential full matrices, and so on (see Figure 5). This forces the matrices to be generated in blocks of 128, which cover 20,480 addresses. The generation of 128 matrix blocks is performed until all the addresses are scheduled. In total, 104,863 matrices are generated.

4.3 Results

Given that the page channels are known, but not their order or phase, page packets must be sent in a redundant manner to guarantee reception within the page scan interval. Therefore, if the phase of a targeted device is not known, the timing of a page scan window occurrence is completely unknown to the sending device. Thus, it is necessary to transmit the same sequence of N_{upw} packets for a duration equal to $T_{page\,scan}$ to guarantee that regardless of when

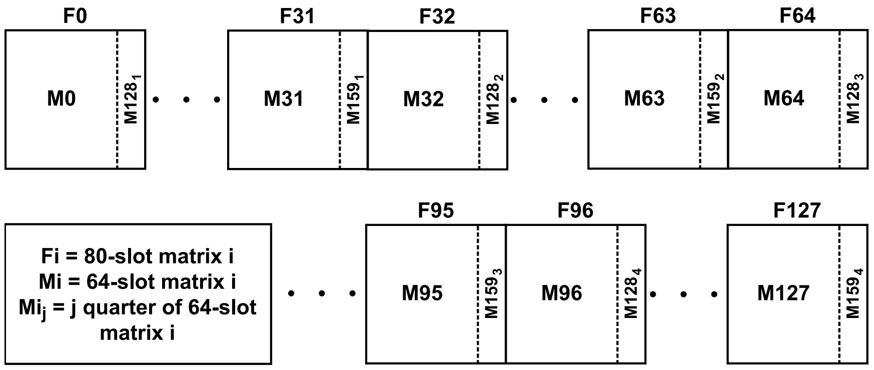

Figure 5. 128 matrix block.

the listening device wakes up it will receive all N_{upw} packets required for a successful brute force search.

The average time taken for a brute force search of an address (T_{bf}) can be calculated as:

$$T_{bf} = \frac{T_{page\,scan} \times f(N_a,\, N_d,\, N_{upw})}{2}.$$

N_a represents the number of addresses to be guessed and N_d is the number of devices used in the brute force search. The function f is determined by the packet schedule and represents the number of matrices needed to search for all the addresses. The denser the matrices, the lower the f value and the lower the T_{bf} value.

Table 1. Average time required for brute force search.

Mode	$T_{page\,scan}$	T_{bf}
R0	11.25 ms	19.66 min
R1	1.28 s	18.64 hrs
R2	2.56 s	37.28 hrs

The Bluetooth specification defines three page scan modes, R0, R1 and R2, which differ only in their page scan intervals. Table 1 lists the average times to find addresses for the three modes of operation. In the best case (mode R0), an address is found in under 10 minutes. However, the majority of Bluetooth devices operate in mode R1. A sample calculation for mode R1 is given by:

$$\begin{aligned} T_{bf} &= \frac{1.28\,s \times f(2^{24},\ 79,\ 80)}{2} \\ &= \frac{1.28\,s \times 104,863\ matrices}{2} \\ &= 18.64\ hours. \end{aligned}$$

4.4 Practical Considerations

The theoretical results presented above must be considered in the context of real-world implementations. The principal issues relate to the target's processing speed, channel interference and overall scalability.

The target's ability to process page packets may not be fast enough to keep up with the speed at which it receives packets from the brute forcing devices. This is a consequence of T_{delay} being significantly shortened. Currently, it is unknown if most Bluetooth devices can process packets rapidly enough to accommodate the new T_{delay}. Of course, T_{delay} can be increased as required, but the overall time required for a brute force search is also increased.

Channel interference can be quite significant in real-world environments. The interference could come from the brute forcing devices themselves or from environmental noise (including other Bluetooth devices). Packet scheduling ensures that the brute forcing devices do not interfere with each other because each device transmits on its own channel. Environmental noise is more difficult to mitigate; future research should attempt to properly model the noise and identify possible mitigation strategies. Since packet scheduling is never able to completely fill all the packet matrices, the potential exists to utilize unused matrix slots for redundant address searches. A trade-off procedure could be developed that balances the number of unique guesses (and total search time) with the number of redundant guesses that could be adjusted based on the noise model.

Finally, with respect to scalability, it may be feasible to use 79 devices, but this may not be practical. It is essential that the model be capable of scaling down to fewer than 79 devices. The time required for a brute force search scales linearly with the number of devices used. When one device is available, everything is done the same way as with 79 devices, except that only one channel is transmitted per address space search. Page packets would then be sent for all addresses on channel 1, then channel 2, etc. Similarly, in the case of two devices, two channels could be utilized simultaneously (and so on).

5. Conclusions

Mobile phones, PDAs, input devices, smart card readers, computers and automobiles have all become Bluetooth-enabled devices. More than 1 billion Bluetooth devices are in use worldwide, and another 13 million devices are shipped each week [9]. The ability to glean information from the devices is a serious concern; due to the ubiquity of Bluetooth devices, this represents a threat to the critical infrastructure.

The non-discoverable mode is intended to serve as a protection mechanism for Bluetooth devices. However, the ability to detect non-discoverable devices threatens security and privacy. Detecting a device is a precursor to tracking its location; once the device is located, other exploits can be launched.

The methodology presented in this paper drastically reduces the time needed for a brute force attack on Bluetooth devices. The combination of collision avoidance, multiple scanning devices and condensed packet timing enables the average device discovery time to be reduced to a mere 20 minutes.

This work opens several avenues for future research. One extension is to verify the theoretical results via a software simulation using a coded extension to IBM's BlueHoc 2.0 [8]. Other extensions include designing a hardware solution for adjusting detection parameters based on environmental variables, refining the packet scheduling technique, and reducing the address search space via a statistical analysis of assigned MAC addresses. Of course, the most pressing research issue is to devise mitigation strategies that will render Bluetooth devices immune to brute force attacks.

Acknowledgements

The authors gratefully acknowledge the support of Professor Gerald Masson and the suggestions provided by Jeff Cua.

References

[1] Bluetooth Special Interest Group, Bluetooth core specification v2.0 + EDR (bluetooth.com/Bluetooth/Learn/Technology/Specifications), 2004.

[2] Bluetooth Special Interest Group, Wireless security (www.bluetooth.com /Bluetooth/Learn/Security), 2007.

[3] C. Gehrmann, Bluetooth security white paper, Bluetooth SIG Security Expert Group (grouper.ieee.org/groups/1451/5/Comparison%20of% 20PHY/Bluetooth_24Security_Paper.pdf), 2002.

[4] K. Haataja, Two practical attacks against Bluetooth security using new enhanced implementations of security analysis tools, *Proceedings of the IASTED International Conference on Communication, Network and Information Security*, pp. 13–18, 2005.

[5] J. Hallberg, M. Nilsson and K. Synnes, Bluetooth positioning, *Proceedings of the Third Annual Symposium on Computer Science and Electrical Engineering*, 2002.

[6] M. Herfurt, and C. Mulliner, Remote device identification based on Bluetooth fingerprinting techniques, White Paper (version 0.3) (trifinite.org /Downloads/Blueprinting.pdf), 2004.

[7] IEEE Registration Authority, Public OUI listing (standards.ieee.org/reg auth/oui/index.shtml), 2006.

[8] A. Kumar, BlueHoc: Bluetooth performance evaluation tool (bluehoc.sou rceforge.net).

[9] M. Lev-Ram, Bluetooth's amazing makeover, *Business 2.0*, June 14, 2007.

[10] F. Wong and F. Stajano, Location privacy in Bluetooth, *Proceedings of the Second European Workshop on Security and Privacy in Ad Hoc and Sensor Networks (LNCS 3813)*, R. Molva, G. Tsudik and D. Westhoff (Eds.), Springer-Verlag, Berlin-Heidelberg, pp. 176–188, 2005.