Relay Attacks on Bluetooth Authentication and Solutions

Albert Levi[1], Erhan Çetintaş[2], Murat Aydos[3],
Çetin Kaya Koç[4], and M. Ufuk Çağlayan[5]

[1] Sabanci University, Fac. of Eng. & Nat. Sci., Orhanli, Tuzla, TR-34956, Istanbul, Turkey
levi@sabanciuniv.edu
[2] TUBITAK – UEKAE, National Research Institute of Electronics and Cryptology
Gebze, TR-41470, Kocaeli, Turkey
cetintas@uekae.tubitak.gov.tr
[3] Pamukkale University, Dept. of Computer Engineering, Denizli, TR-20020, Turkey
maydos@pamukkale.edu.tr
[4] Oregon State Univ., School of Electr. Eng. & Comp. Sci., Corvallis, OR 97331 USA
koc@ece.orst.edu
[5] Boğaziçi University, Dept. of Computer Engineering, Istanbul, TR-34342, Turkey
caglayan@boun.edu.tr

Abstract. We describe relay attacks on Bluetooth authentication protocol. The aim of these attacks is impersonation. The attacker does not need to guess or obtain a common secret known to both victims in order to set up these attacks, merely to relay the information it receives from one victim to the other during the authentication protocol run. Bluetooth authentication protocol allows such a relay if the victims do not hear each other. Such a setting is highly probable. We analyze the attacks for several scenarios and propose practical solutions. Moreover, we simulate attacks to make sure about their feasibility. These simulations show that current Bluetooth specifications do not have defensive mechanisms for relay attacks. However, relay attacks create a significant partial delay during the connection that might be useful for detection.

1 Introduction and Background

Bluetooth [1] is a promising short-range radio link technology for wireless connectivity of portable electronic devices, such as mobile phones, laptop computers, palm computers and digital cameras. The Bluetooth system operates in the 2.4 GHz ISM (Industrial Scientific Medicine) band. In order to avoid interference with other piconets (piconet is Bluetooth's personal/local area network) and/or other devices using the ISM band, the master of a piconet synchronizes its slaves to hop among several RF channels in a pseudo-random sequence.

Bluetooth specification defines link level security mechanisms to provide confidentiality, integrity and authentication between Bluetooth devices. However, there are some vulnerabilities in the Bluetooth security as proposed in [2, 3, 4].

In this paper, we point to relay attacks on Bluetooth authentication protocol. In relay attacks, the attacker places itself in two distinct piconets and picks two victims, one in each piconet. The attacker impersonates those victims by forwarding

C. Aykanat et al. (Eds.): ISCIS 2004, LNCS 3280, pp. 278–288, 2004.

authentication messages generated by one of them to another between the piconets. As opposed to the man-in-the-middle attacks described in [2], the attacker does not need to know any shared secret between the victims in order to set up our relay attacks. We simulate relay attacks to assess their feasibility. Moreover we use simulation to evaluate the delays caused by the attack and to see if these delays could be used as a detection mechanism. We propose two other low-cost solutions as well.

The rest of Section 1 gives an overview of Bluetooth key management and authentication scheme. Relay attacks are explained in Section 2. Mechanisms to detect relay attacks are proposed in Section 3. Simulation results are presented in Section 4. Conclusions and some discussions are in Section 5.

1.1 Key Management and Authentication in Bluetooth

There are several key types in Bluetooth, but the attacks described here depend on the initialization and combination keys. Initialization key (K_{init}) is calculated at both sides of communication using a pre-shared PIN, a random number and a Bluetooth Device Address (BD_ADDR). K_{init} is used to exchange the *Combination Key*, which is one of the members of the Bluetooth "Link Key" family. These keys are used for authentication. Both ends of communication, say *A* and *B*, contribute to the combination key (K_{AB}) in a secure way by encrypting some random numbers. Current link key or K_{init} is used as the key for this encryption. Link keys are stored in Bluetooth devices and they are reused whenever necessary.

Bluetooth uses a simple challenge-response authentication scheme. The verifier sends a 128-bit random number called AU_RAND to the claimant. Claimant calculates the authentication response called SRES, which is a cryptographic function of AU_RAND, its own BD_ADDR, and the current link key. Claimant sends SRES to the verifier. Meanwhile the verifier computes the same SRES and checks whether the computed one is equal to the received one. If so, that means the claimant is really who it claims to be.

2 Relay Attacks

In this section, we describe relay attacks proposed in the paper. In the relay attacks, adversary *C* talks to victim *A* posing as victim *B*, and to *B* posing as *A*. All authentication messages that *C* needs are generated by real *A* and *B*. *C* conveys these messages from *A*/*B* to *B*/*A*. We present two types of relay attacks: (i) two-sided, and (ii) one-sided. In a two-sided relay attack, both victims are impersonated. In a one-sided attack, only one victim is impersonated.

In [2], some man-in-the-middle and impersonation type of attacks are proposed where the attacker knows or can guess the PIN or existing link key between victims. Relay attacks are similar to man-in-the-middle attacks. There exists an adversary located between the sender and receiver, but the only activity of the adversary is to relay information that it receives from one to another without changing the content. Unlike the attacks in [2], the adversary does not need to know a shared secret.

2.1 Special Conditions of Bluetooth and Attack Settings

Relay attacks are possible if:
(i) actual communication between the real sender and receiver is disconnected and they cannot listen to each other anymore,
(ii) network infrastructure does not have a global infrastructure for routing and locating its users, and
(iii) adversary is capable enough to impersonate each of the victims to the other, even if the victims are located in distant locations.

As an example attack setting, suppose the victims *A* and *B* have communicated for some time and then terminated the communication. They may easily end up in different locations due to their mobile and ad-hoc behavior. In this example, we assume two users working in the sales department of a Bluetooth-enabled office. These two users exchange their data using a service configured in Bluetooth Security Mode 3 that requires only authentication. It is assumed that no encryption and application layer security are employed for this service. The users' laptops are normally part of a piconet within their department, but whenever one of them moved to conference room for a meeting, it becomes a part of the piconet in the conference room. Even if they might be close to each other, they cannot listen to each other since every piconet has a different frequency hopping order. Once the attacker impersonates the users, it can, for example, transfer fraudulent data or alter sales reports.

Although there are valuable efforts in the literature for forming and routing for Bluetooth Scatternets [7, 8, 9], the short-range characteristics of Bluetooth devices would not enable to have a Bluetooth-based global ad hoc network, thus satisfying the feasibility of the conditions (i) and (ii) above. One may argue that some application layer Bluetooth profiles, such as IP over Bluetooth, could provide global connectivity. However, such applications should be implemented over L2CAP (Logical Link Control and Adaptation Protocol) and consequently the LMP (Link manager Protocol) layers of Bluetooth at which relay attacks are implemented. Thus, such global connectivity does not help to avoid relay attacks since packets used in the relay attacks remain local and do not pass through the gateways.

In order to satisfy condition (iii), the adversary, *C*, should contain two different Bluetooth units, *Ac* and *Bc* (*Ac* and *Bc* denote *A* and *B* impersonated by *C*), with adjustable BD_ADDRs. *Ac* should be located close to *B*, and *Bc* close to *A*. The adversary *C* is also equipped with a special communication interface between *Ac* and *Bc*. This interface is not necessarily a Bluetooth interface; actually Bluetooth is not useful for communication between *Ac* and *Bc* if they are far apart. Some other wireless or wired methods can be used for communication between distant *Ac* and *Bc*. Moreover, *C* should know the pseudo-random frequency hopping order of real *A* and *B* in order to eavesdrop on their communication. Jakobsson and Wetzel proposed a method to determine this order in [2].

Each Bluetooth device is identified using an overt and fixed Bluetooth Device Address (BD_ADDR). It is embedded in the device and normally not changeable. However, a hostile manufacturer can build a Bluetooth device with an adjustable BD_ADDR. With the current trend of increased Bluetooth deployment in almost all type of mobile devices, attacks on Bluetooth may create a spying market and such manufacturers may come into sight.

2.2 Two-Sided Relay Attack

This attack is shown in Figure 1. Here, *C* must wait for a real request for connection from either *A* or *B*. Suppose real *A* wants to establish connection to *B*. The connection establishment process starts with the paging procedure. *A* first pages *Bc* thinking that it is real *B*. After the paging procedure, *A* sends *LMP_host_connection_req* command to *Bc*. *Bc* accepts the connection request by sending back *LMP_accepted*. Meanwhile, *Ac* pages *B* and initiates a connection establishment procedure posing as *A*.

The above connection establishment process is valid if victims *A* and *B* are distant. If they are close to each other, in order *A* to connect *Bc* instead of *B*, the attacker's *Bc* interface must respond to the paging request faster than *B*. The details of such a setting are described by Kügler [4]. In addition, Kügler [4] also discusses that the attacker's *Ac* interface must use a clock value different from the clock of *A*. Thus, both *A* and *B* use the same frequency hopping order with different offsets and do not hear each other.

The current link key between *A* and *B* may or may not be changed at each connection. The attacker does not need to know this key. Thus, changing the link key or using the current one do not cause any problem in attack setting. If the link is to be changed, then the next step is the exchange of combination key contributions (the random numbers which are encrypted by XORing with the current link key). *A* sends its encrypted random number, $RAND\_NR_A$, to *Bc* in an *LMP_comb_key* command. *C,* using its *Ac* interface, relays this encrypted random number to real *B* as if it is sent by *A*. After receiving this number, *B* sends out its encrypted random number $RAND\_NR_B$ to *Ac*, and *C* forwards it to real *A* while wearing its *Bc* hat. After these message rounds, both real *A* and real *B* compute the same combination key K_{AB} and this key is assigned as new link key.

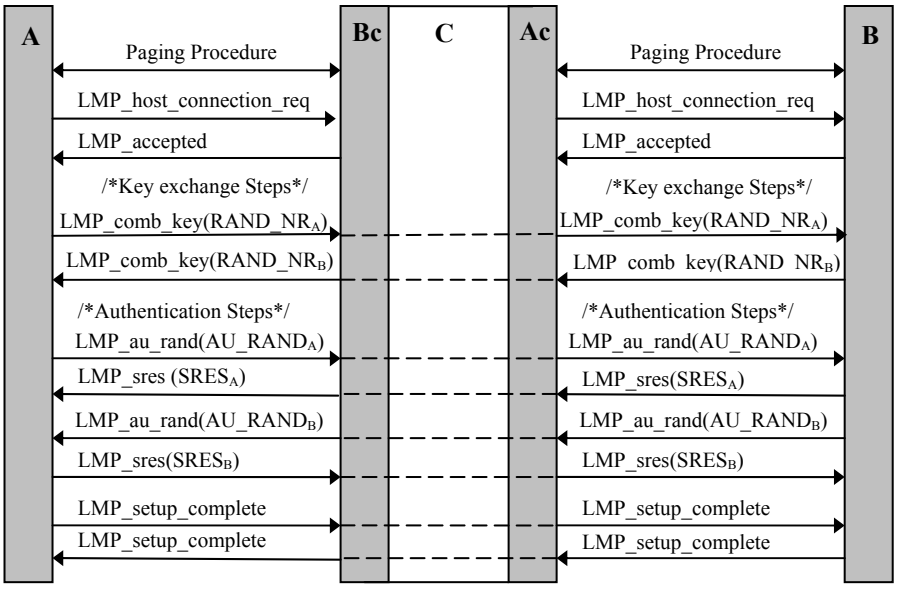

Fig. 1. Two-sided relay attack

The next steps are for authentication. *A* sends *LMP_au_rand* command to *Bc* (thinking that it is *B*) along with a 128 bit random number called AU_RAND_A. After sending AU_RAND_A, *A* expects the corresponding authentication response $SRES_A$. *Bc* cannot calculate $SRES_A$, since it does not know the current link key, but *C* (using its *Ac* interface) can forward AU_RAND_A to real *B* in another *LMP_au_rand* command as if *A* requests authentication of *B*. The response of real *B* to this command is an *LMP_sres* command that contains $SRES_A$. *C* forwards $SRES_A$ to *A* in another *LMP_sres* command as the authentication response of *Bc*. After that *A* thinks that *B* is authenticated, but the truth is that *Bc* is authenticated. Similar steps are taken in the case of mutual authentication where *B* requests authentication of *Ac* thinking that it is *A*. At the end, both *A* and *B* think that they authenticated each other, but the fact is that *C* impersonated both of them. *C* exploits both real *A* and *B* to generate authentication responses $SRES_B$ and $SRES_A$.

If there is no existing link key established beforehand or the link key is somehow unavailable (e.g. lost, compromised, expired, etc.), then *A* and *B* should initiate K_{init} generation before combination key generation steps. Two-sided attack works in such a setting too, because the attacker would only need to relay some messages as in combination key generation steps.

2.3 One-Sided Relay Attack

The adversary *C* can make use of this attack by initiating communication with one of the victims impersonating the other one. This attack is possible only when the victims can be convinced to use the existing link key.

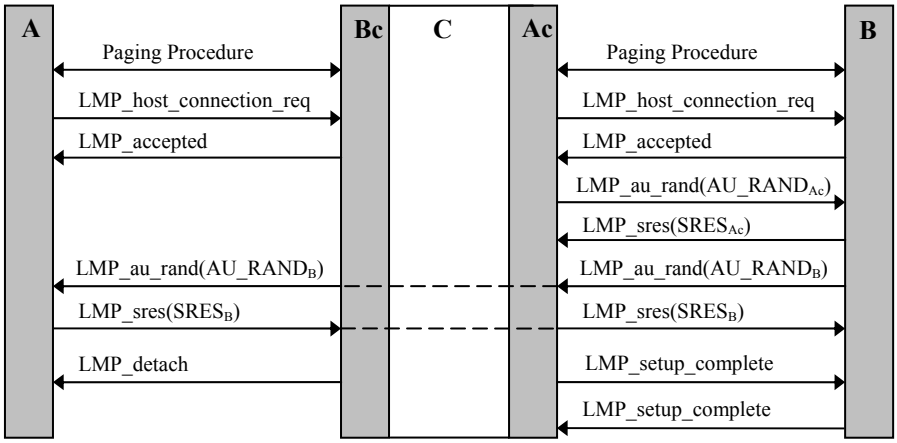

Fig. 2. One-sided relay attack

Figure 2 depicts one-sided attack. In this attack, *C* impersonates *A* to talk to *B*. We assume that real *A* and *B* already have a link key established. Communication is requested by *C* (*Ac*). In order to make new connection *Ac* first pages *B* and then sends *LMP_host_connection_req* command to *B*. *B* accepts the connection request by sending back *LMP_accepted*. At the same time *Bc* pages *A* and starts the connection

establishment procedure with *A*. After the first step, *Ac* sends *LMP_au_rand* to *B*. In this command *Ac* includes a dummy random number AU_RAND_{Ac}. *B* wrongfully thinks that real *A* requests authentication and sends back the corresponding authentication response $SRES_{Ac}$. *Ac* has no way to check the correctness of $SRES_{Ac}$, so it implicitly assumes that real *B* is indeed genuine. Having sent $SRES_{Ac}$, *B* sends its authentication challenge AU_RAND_B to *Ac*. *Ac* should obtain $SRES_B$, which is the SRES corresponding to AU_RAND_B, so *C* sends out AU_RAND_B to real *A* using its *Bc* interface. Real *A* mistakenly thinks that *B* requests authentication, and calculates and sends $SRES_B$ to *Bc*. Then *Ac* forwards $SRES_B$ to real *B*. The connection setup is completed by mutually sending *LMP_setup_complete*. These steps authenticate *Ac* to *B* as if *Ac* were the real *A*. At the end *Bc* sends an *LMP_detach* command to end its communication with real *A*, since *A* is not needed anymore.

Although the above attack explanation is for distant victims, it also works for close ones by using different clock values as explained in Section 2.2 for two-sided attack.

3 Proposed Solutions

In this section, we propose three practical solutions to detect relay attacks.

3.1 Solution 1: For Victims in Close Piconets

One method for preventing the relay attacks is to include unforgeable piconet-specific information in SRES calculation. Such information could be hop sequence parameters, channel access code, which is added to each packet sent within the piconet, and the sync word, which is part of the channel access code. Unfortunately all of them are based on LAP (lower address part) of master's BD_ADDR and/or master's clock. Since the attacker is the master in one of the piconets, it can enforce those parameters that are learned from the other piconet where the attacker is a slave. The only exception is when these two piconets are close to each other. Channel access code, sync word, hop sequence and phase cannot be the same for those piconets due to interference problems. That means piconet specific information based relay attack control works for close piconets.

Implementation of this control is easy. It is sufficient to consider master's clock and LAP values in SRES calculation. To do so, the least significant 42 bits of the AU_RAND values could be XORed with the concatenation of clock and LAP values at each piconet for SRES calculation and verification. Real *A* and *B* use different clock and/or LAP values since the attacker cannot enforce the same values, because otherwise messages of two piconets mix up. The updated authentication mechanism is shown in Figure 3. The original Bluetooth authentication scheme does not have the XOR part, i.e. AU_RAND is directly fed into E1 boxes.

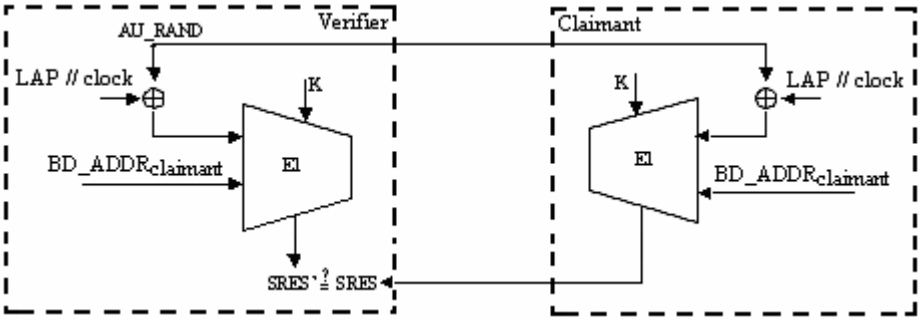

Fig. 3. Updated Bluetooth challenge-response authentication scheme that is sensitive to piconet (master) clock and LAP

3.2 Solution 2: For One-Sided Relay Attack

In the original Bluetooth scheme, mutual authentication is performed exclusively between master and slave. First, one is authenticated with AU_RAND (challenge) and SRES (response) exchange. Then the other is authenticated again using a challenge/response mechanism. We propose to change this authentication message exchanges in a nested form such that first both parties exchange their AU_RAND values and claimant does not send its SRES before getting the legitimate SRES from the verifier. This message exchange is shown in Figure 4.

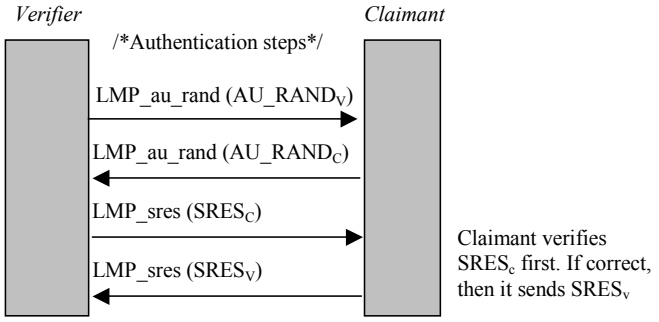

Fig. 4. Nested mutual authentication

In this method, which is effective against one-sided attack, the attacker cannot obtain SRES values from the victims, since both victims first wait for the SRES value from the other party (i.e. from the attacker). Since the attacker acts as a verifier in both piconets, its authentication challenge is responded with another authentication challenge from the genuine entities; SRES values are not sent and protocol eventually times out. Unfortunately, nested mutual authentication does not solve the problems associated with two-sided attacks. One of the victims is the verifier in that scenario, so the attacker can obtain the SRES from it.

3.3 Solution 3: Variance in Delays

This solution is based on consideration of variance in end-to-end delays between normal connection cases and attack cases, as will be discussed in Section 4.

4 Simulation Results

In order to analyze the effectiveness of the proposed relay attacks, we developed a simulator using C++ programming language that simulates baseband link connection and authentication procedures according to the baseband, security and LMP specifications.

First, we implemented the attack scenarios in our simulation environment and Link Manager transactions of the simulated Bluetooth devices are compared in normal connection and attack cases. During this analysis, we have realized that there is absolutely no difference in terms of transaction outputs between attacked victims and a non-attacked ones. Thus, we conclude that victim devices cannot be aware of the relay attacks by checking connection establishment transactions. These transaction outputs are not shown here because of space restrictions.

4.1 Timing Analysis

In our attack scenarios, one victim waits for receiving the authentication response SRES while the attacker is getting this SRES from the other victim. If this duration is too much, LMP response timeout may exceed. In addition, due to relay attacks the connection establishment process may take long time and one of the victims may be aware of the attack. Thus, in our experiments we measure the connection establishment time and the latency in receiving LMP_SRES. Here only the timings of successful transmissions are taken into account. In case of retransmissions, which are probable in Bluetooth, the baseband layer should inform link manager so that the corresponding timers are reset.

We first measured connection establishment times during normal connection cases and attack cases. Particular increase has been noticed in the connection time of an attacked victim as compared to non-attacked one. However, as discussed in [5], connection time can vary between devices which are produced by different manufacturers or whose clocks are not initially well synchronized. Thus we conclude that the increase was not big enough in order to conclude that connection time increase could be used as an attack detection mechanism.

In our experiments, we also considered how our relay attacks affect the duration between sending LMP_AU_RAND command and receiving LMP_SRES response in victim devices *A* and *B*. Figure 5 shows a histogram of the latency in receiving LMP_SRES in the normal connection establishment. *A* obtains the link key from *Host A* before receiving LMP_AU_RAND from *B*. However, *B* gets the link key after receiving LMP_AU_RAND from *A*. Therefore, *A* waits longer than *B* to receive LMP_SRES. The average waiting time is 10.378 ms for *A* and 2.008 ms for *B*.

Figure 6 and Figure 7 show the histograms of the waiting times for receiving LMP_SRES response in two-sided and one-sided relay attacks.

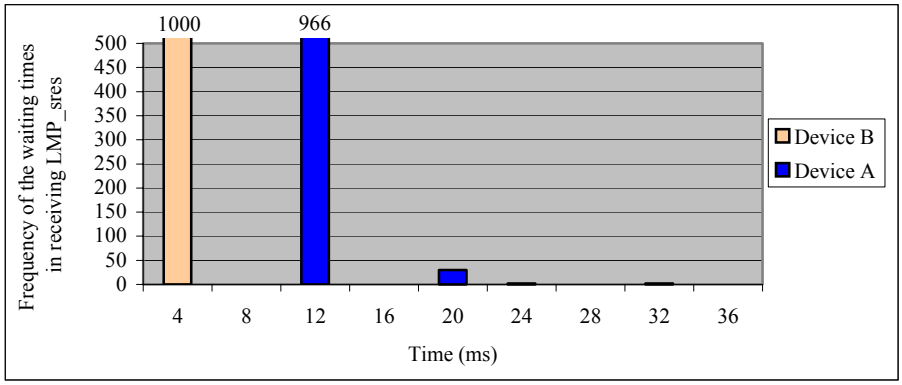

Fig. 5. Histogram for the waiting times for receiving LMP_sres in the normal connection

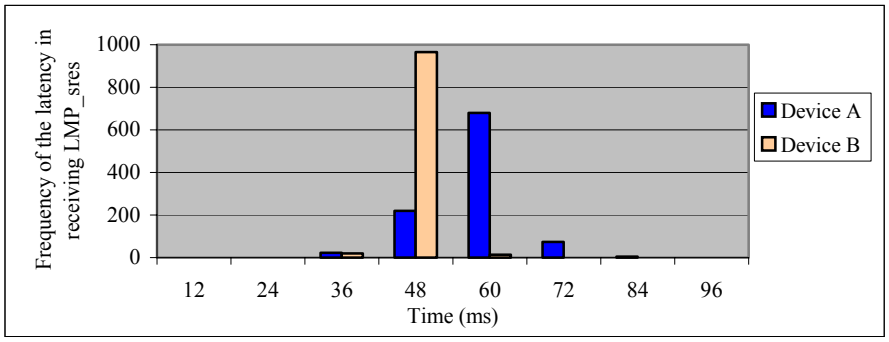

Fig. 6. Frequency of the latency in receiving LMP_sres in two-sided relay attack

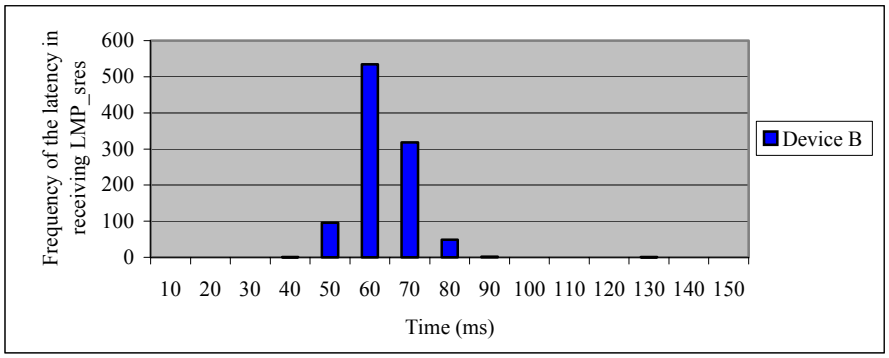

Fig. 7. Frequency of the latency in receiving LMP_sres in one-sided relay attacks

In two-sided relay attack, the average latency in receiving LMP_SRES for *A* and *B* is 50.978 and 39.625 ms respectively. As we see in Figure 7, most of the waiting times are between 50 ms and 80 ms (on average 62.578 ms) in one-sided relay attack. Thus we conclude that relay attacks increase the latency in receiving LMP_SRES response; two-sided relay attack increases this latency 5 times in *A* and 20 times in *B*. Similarly, the latency in one-sided relay attack is 30 times more than the latency in the normal connection. According to Bluetooth Link Manager Protocol specification, the time between receiving an LMP PDU and sending a valid response PDU must be less than the LMP response timeout, which is 30 seconds. Therefore, the LMP response timeout does not expire due to relay attacks. However, if enough intelligence is added to LMP protocol, the victims may detect the relay attacks by checking the considerable increase in latency of getting LMP_SRES response. For example, one device measures average delays in connection establishment processes and stores these values in its memory. During each connection, it estimates the current delay by considering a sequence of previous delays and then compares the estimated delay with new measured one. If there is a major difference, then the device may decide that there is an attack going on. One possible method for estimating the current delay would be simply to compute arithmetic average of stored delays. Since it is not necessary to store all past delays, this method is appropriate for devices with limited memory resources. Another method for the current delay estimation would be to compute exponential average delay by using the smoothing procedure as in the TCP protocol. With this method, we can give the most recent measurements a greater weight than the older ones. Dynamically estimating the current delay reduces the impact of transmission errors in decisions about the relay attacks.

5 Conclusion and Discussions

We present two important relay attacks on Bluetooth authentication method for impersonation purposes. The adversary need not obtain any secret (like PINs or current keys) of the victims. He/she simply relays some protocol messages from one victim to another without alteration.

Relay attacks are to make fail only Bluetooth authentication, not encryption. The attacker cannot continue its attack if the victims prefer to have encrypted communication. However, in Bluetooth specification [1], having no encryption is a valid option, and during negotiation the adversary can indeed convince the victims not to have encrypted communication. Bluetooth authentication is performed to authenticate entities, not messages, mostly for access control decisions. Traditionally access control does not require encrypted communications, once the access is granted. Thus, message encryption and entity authentication need not coexist all the time. As suggested in [6], it is conceivable that a device might want to perform only-authentication because it was not using encryption on the link, but it still wants to check if it is communicating with the correct device. The processing power limitations of a device might not let it use encryption that requires constant processing. However, authentication is once-per-session and can be tolerated even for restricted devices.

Relay attacks are based on a deception: both victims think they are in the same piconet. However, they are not. They are actually in different piconets. If the victims can include some information about their actual piconets in SRES, then relay attacks could be detected. As discussed in this paper, such piconet-specific information is unfortunately forgeable by the attacker, if the piconets are not close to each other. If they are close, then inclusion of LAP (lower address part) of the master BD_ADDR and master clock in SRES messages solves the problem. Such a solution is of limited use, but does not cause a remarkable load on the entities; the extra processing is just an XOR computation. Another limited use, but efficient precaution could be to exchange the challenge messages (AU_RAND) before sending out the responses (SRES). The claimant waits for the SRES for its challenge first. In this way, it does not give out the SRES to be relayed. This solution works only if the attacker is the verifier in both piconets. This situation corresponds to the one-sided attack described in this paper.

In the simulations of the attacks, we have realized that the victims cannot detect relay attacks if they strictly follow Bluetooth specifications. On the other hand, our analysis of the simulation results demonstrates that there is a perceptible variation in some end-to-end delays between the normal and the attacked connections. One device can estimate the current delay by observing the pattern of delay for recent connection establishments, and then compare the estimated delay with new measured one. A significant increase means that there may be an attack going on. Simple average or exponential average methods would be used for the current delay estimation. Such an intelligent adaptive mechanism can be incorporated in Bluetooth connection establishment procedure at LMP level to determine both types of relay attacks in a low-cost and effective way.

References

1. Bluetooth SIG. Specification of the Bluetooth System – Bluetooth Core Specification, Vol. 0-3, Version 1.2. http://www.bluetooth.org. (2003)
2. Jakobsson, M., Wetzel, S.: Security Weaknesses in Bluetooth. Lecture Notes in Computer Science, Vol. 2020. Springer-Verlag, (2001) 176–191
3. Vainio. J.T.: Bluetooth Security. http://www.niksula.cs.hut.fi/~jiitv/bluesec.html. (2000)
4. Kügler, D.: Man in the Middle Attacks on Bluetooth. Lecture Notes in Computer Science, Vol. 2742. Springer-Verlag, (2003) 149–161
5. Welsh, E., Murphy, P., Frantz, P.: Improving Connection Times for Bluetooth Devices in Mobile Environments. International Conference on Fundamentals of Electronics, Communications and Computer Sciences (ICFS). Tokyo, Japan (2002)
6. Bray, J., Sturman, C.F.: Bluetooth: Connect Without Cables. Prentice-Hall, (2000)
7. Bhagwat, P., Segall, A.: A Routing Vector Method (RVM) for Routing in Bluetooth Scatternets. In: IEEE International Workshop on Mobile Multimedia Communications (MoMuC'99). (1999)
8. Wang, Z., Thomas, R.J., Haas, Z.: Bluenet - A New Scatternet Formation Scheme. In: 35th Annual Hawaii International Conference on System Sciences. (2002)
9. Kapoor, R., Gerla, M.: A Zone Routing Protocol for Bluetooth scatternets. In: WCNC. (2003)