http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 6 (4): 478-489, 2007
ISSN 1812-5638

IPSec Based Bluetooth Security Architecture

Al-Muthanna Al-Hawamdeh, Adil Khan and M. Sikander Hayat Khiyal
Department of Computer Science, International Islamic University, Pakistan

Abstract: Bluetooth has recently obtained an unprecedented success in gaining a wide industry support. The main aim of Bluetooth is to eliminate interconnection cables and to connect one device to another via a universal radio link. First generation Bluetooth chipsets are limited in range to approximately 10 m under typical line-of-sight conditions. Future implementations will provide roughly ten times that range. In the present study, a performance assessment, processing delay and throughput of IPSec over Bluetooth based on the Bluetooth Network Encapsulation Protocol (BNEP) operation scenario, is presented. In particular, the performance of the point-to- point link and the effectiveness of IPSec (authentication and encryption) algorithms in Bluetooth. Moreover, some networking issues that may limit Bluetooth applicability in some environments will be pointed out.

Key words: Bluetooth, IPSec, security, BNEP protocol, MD-5, SHA-1, DES, 3DES

INTRODUCTION

Bluetooth was initially designed as an efficient cable replacement technology primarily for handheld devices. Indeed, all the devices belonging to one person can form a PAN (Personal Area Network) using Bluetooth. Bluetooth protocol stack is divided into five layers (Bluetooth, 2001; Saarinen, 2000):

- Bluetooth Core Protocols: including Baseband, LMP, L2CAP and SDP, comprise exclusively Bluetooth-specific protocols developed by the Bluetooth SIG that are required by most of the Bluetooth devices.
- Cable Replacement Protocol: i.e., RFCOMM protocol is based on the ETSI TS 07.10 that emulates serial line control and data signals over Bluetooth Baseband to provide transport capabilities for upper level services.
- Telephony Control Protocols: including TCS Binary and AT-commands are used to define the call control signaling, mobility management procedures and multiple usage models for the Bluetooth devices to establish the speech and data calls and provide FAX and modem services.
- Adopted Protocols: including PPP, UDP/TCP/IP, WAP, WAE, etc. Due to the open nature of the Bluetooth specification, additional protocols (e.g., HTTP, FTP, etc.) can be accommodated in an interoperable fashion.
- Host Controller Interface (HCI): i.e., the boundary between hardware and software provides a uniform command interface to access capabilities of hardware, e.g., Baseband controller, link manager, control and event registers.

The layers of Cable Replacement, Telephony Control and Adopted Protocols form the application-oriented protocols that enable applications to run over the Bluetooth core protocols.

IPSec

The IPSec protocol suite is used to provide privacy and authentication services at the IP layer. It provides a set of security algorithms plus a general framework that allows a pair of communicating entities to use whichever algorithms provide security appropriate for the communication.

The elements describing the set of IPSec protocols are divided into six groups:

- There is the main Architecture, which broadly contain the general concepts, security requirements, definitions and mechanisms defining IPSec technology.
- There is the ESP Protocol and an AH Protocol.
- The Encryption Algorithm, describes how various encryption algorithms are used for ESP.
- The Authentication Algorithm describes how various authentication algorithms are used for both ESP and AH.

Corresponding Author: Al-Muthanna Al-Hawamdeh, Department of Computer Science, International Islamic University, Pakistan
Tel: (0092) 03335417855

- The Key Management .
- The DOI contains values needed for the other elements to relate to each other. This includes for example encryption algorithms, authentication algorithms and operational parameters such as key lifetimes.

IP AUTHENTICATION HEADER (AH)

The IP Authentication Header (AH) is used to provide connectionless integrity and data origin authentication for IP datagram's and to provide protection against replays.

AH may be employed in two ways: transport mode or tunnel mode. The former mode is applicable only to host implementations and provides protection for upper layer protocols. Tunnel mode may be employed in either hosts or security gateways.

The first step of integrity protection is to create a hash by using a keyed hash algorithm, also known as a Message Authentication Code (MAC) algorithm. A standard hash algorithm generates a hash based on a message, while a keyed hash algorithm creates a hash based on both a message and a secret key shared by the two endpoints. The hash is added to the packet and the packet is sent to the recipient. The recipient can then regenerate the hash using the shared key and confirm that the two hashes match, which provides integrity protection for the packet. IPSec uses Hash Message Authentication Code (HMAC) algorithms, which perform two keyed hashes. Examples of keyed hash algorithms are HMAC-MD5 and HMAC-SHA-1. Another common MAC algorithm is AES Cipher Block Chaining MAC-AES-XCBC-MAC-96- (Kent and Atkinson, 2004).

ENCAPSULATING SECURITY PAYLOAD(ESP)

The Encapsulating Security Payload (ESP) header is designed to provide a mix of security services in IPv4 and IPv6. ESP may be applied alone, in combination with the IP Authentication Header (AH), or in a nested fashion (Kent and Atkinson, 2004b).

ESP is used to provide confidentiality, data origin authentication, connectionless integrity, an anti-replay service and limited traffic flow confidentiality. The set of services provided depends on options selected at the time of Security Association establishment and on the placement of the implementation.

ESP uses symmetric cryptography to provide encryption for IPSec packets. Accordingly, both endpoints of an IPSec connection protected by ESP encryption must use the same key to encrypt and decrypt the packets. When an endpoint encrypts data, it divides the data into small blocks and then performs multiple sets of cryptographic operations using the data blocks and key. Encryption algorithms that work in this way are known as block cipher algorithms. When the other endpoint receives the encrypted data, it performs decryption using the same key and a similar process, but with the steps reversed and the cryptographic operations altered. Examples of encryption algorithms used by ESP are AES-CBChaining, AES Counter Mode, DES and 3DES (Kent and Atkinson, 2004b).

IPSec OVER BLUETOOTH

The proposed idea is that authentication and encryption in Bluetooth to be provided on IP or application level by using IPSec according to RFC 2401 (1998) at the IP level. A protocol like IPSec is most suitable to secure end-to-end IP services like Virtual Private Network (VPN) services. IPSec can be used for any IP connection independent of the particular access method. Here only LAN access using the Bluetooth wireless technology is considered. It is important to notice that the use of link level security and VPN solutions does not exclude each other but rather complement each other.

IPSec, however, can protect any protocol running above IP and any medium which IP runs over. More to the point, it can protect a mixture of application protocols running over a complex combination of media. This is the normal situation for Internet communication; IPSec is the only general solution.

The problems raises is that Bluetooth enabled devices will have the ability to form networks and exchange

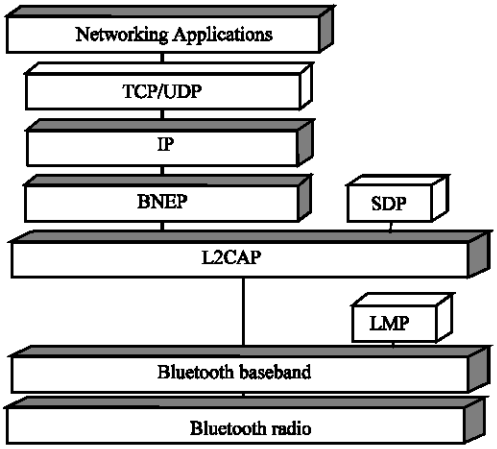

Fig. 1: BNEP Stack

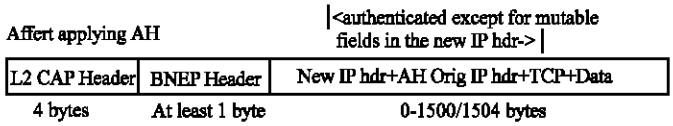

Fig. 2: BNEP after applying AH

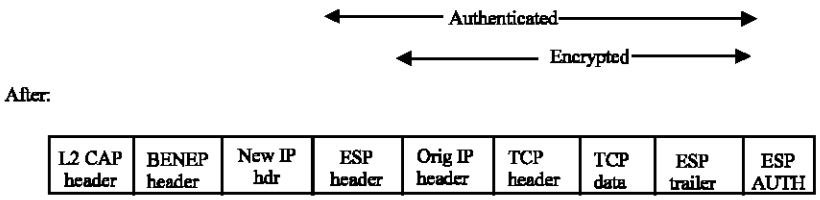

Fig. 3: BNEP after applying ESP

information. For these devices to interoperate and exchange information, a common packet format needs to be defined to encapsulate layer 3 network protocols.

Due to that, a specific packet format used to transport common networking protocols over the Bluetooth media (RFC 894,1996) (RFC 2225,1998) (RFC2734, 1999). The packet format is based on Ethernet/DIX Framing as defined by IEEE 802.3 according to Ether Typ (2006) (Anonymous, 1980) (Inlernat Enigeering Task Force, 1996; 1998; 1999).

The functional requirement for Bluetooth networking encapsulation protocol includes the following according to BNEP specification (2002):

- Support for common networking protocols such as IPv4, IPv6, IPX and other existing or emerging networking protocols.
- Low Overhead -- The encapsulation format SHALL be bandwidth efficient.

The following points illustrate the BNEP header format.

Bnep type: Seven bit Bluetooth Network Encapsulation Protocol.

Type value identifies the type of BNEP header contained in this packet (BNEP specification, 2002).

Extension flag (E): One bit extension flag that indicates if one or more extension headers follow the BNEP Header before the data payload if the data payload exists. If the extension flag is equal to 0×1 then one or more extension headers follows the BNEP header. If the extension flag is equal to 0x0 then the BNEP payload follows the BNEP header (BNEP specification, 2002).

BNEP packet: Based on the BNEP Type (BNEP specification, 2002).

Bluetooth Network Encapsulating Protocol (BNEP) accommodates IP communication by transporting IP packets between two Ethernet-based link layer end-points on an IP segment. It encapsulates the IP packets in BNEP headers, letting the source and destination addresses reflect the Bluetooth end-points and setting the 6-bit Networking Protocol Type field to code for an IP packet in the payload. BNEP finally encapsulates the BNEP packet in an L2CAP header and sends it over the L2CAP connection (Fig. 1).

Figure 2 and 3 show BNEP with an IPv4 packet payload sent using L2CAP after positioning AH header and ESP header for tunnel modes.

CRYPTANALYSIS

In Bluetooth Encryption, several attacks and attempts at cryptanalysis of E0 (Lu and Vaudenay, 2004) and the Bluetooth protocol have been made and a number of vulnerabilities have been found. In 1999, Miia Hermelin and Kaisa Nyberg showed that E0 could be broken in 2^{64} operations (instead of 2^{128}), if a 2^{64} bits output is known. This type of attack was subsequently improved by Kishan Chand Gupta and Palash Sarkar. scott fluhrer, found a theoretical attack with a 2^{80} operations precalculation and a key search complexity of about 2^{65} operations. He deduced that the maximal security of E0 is equivalent to that provided by 65-bit keys and that longer keys do not improve security. Fluhrer's attack is an improvement upon earlier work by golic, bagini and morgani, who devised a 2^{70} operations attack on E0.

Lu and Vaudenay (2005). Published a cryptanalysis of E0 based on a conditional correlation attack. Their best result required the first 24 bits of $2^{23.8}$ frames and 2^{38}

computations to recover the key. The authors assert that "this is clearly the fastest and only practical known-plaintext attack on Bluetooth encrytion compare with all existing attacks (Lu and Vaudenay, 2004). In the authentication scheme of Bluetooth there seems to be some weaknesses.

SAFER+ according to NIST (1998) was submitted as a candidate for the Advanced Encryption Standard and has a block size of 128 bits. The cipher was not selected as a finalist. SAFER+ was included in the Bluetooth standard as an algorithm for authentication and key generation.

Its found that in SAFER+/192 and SAFER+/256, the key schedules do a poor job of getting the whole key involved quickly in the encryption process. SAFER+/192 takes five (of twelve) rounds to get the whole key involved in the encryption process; SAFER+/256 takes nine (of sixteen) rounds to do so. This contrasts with SAFER+/128, where every round is affected by every bit of key.

Due to this slow key diffusion, a meet-in-the-middle attack was found on SAFER+/256. This attack requires work equivalent to about 2^{240} SAFER+/256 encryptions and about $12*2^{24}$ bytes of memory. Also due to this slow key diffusion, a related-key attack was found on SAFER+/256. This attack requires very little memory, $3 * 2^{32}$ chosen plaintexts encrypted under two different keys with a chosen XOR relationship and work approximately equivalent to 2^{200} SAFER+/256 encryptions (Kelsey *et al*., 1999).

SYSTEM DESIGN

The simulation of the IPSec protocols in NS2 was based on the existing implementation of wireless network NS-2 (The Network Simulator) version 2 and UCBT (Bluetooth extension for NS2) (Fig. 4). UCBT implements a full Bluetooth stack, including Baseband, LMP, L2CAP, BNEP layers (UCBT, 2004).

Among the most important design principles for BT networking are:

- Providing for flexibility in usage as a universal short-range low-cost low-power technology in a variety of different scenarios.
- Ensuring that BT devices made by different manufacturers can inter-operate.

Following the first principle, BT networking topology is built upon the flexible concepts of the scatternet and the piconet. A piconet is an ad hoc collection of BT devices, where one of the devices takes the role of the master of the piconet and the other devices take the role of the slaves. Since each node could be a slave on multiple piconets, a larger network structure may be formed out of multiple piconets. This larger structure is the scatternet.

Following the second design principle, a BT protocol stack has been defined (Fig. 5). Usage profiles have also been defined for different usage scenarios, such as LAN access, to allow devices from different manufacturers to inter-operate. The profiles are collections of messages, procedures, features and parameter settings that must be used in order to provide specific services or usage scenarios for BT (Bluetooth, 2001). The use of profiles somewhat limits the flexibility in terms of usage of higher-layer protocols, network topologies and usage scenarios to what is contained in the profiles.

This is a tradeoff between the two principles discussed. In an effort to make use of existing protocols,

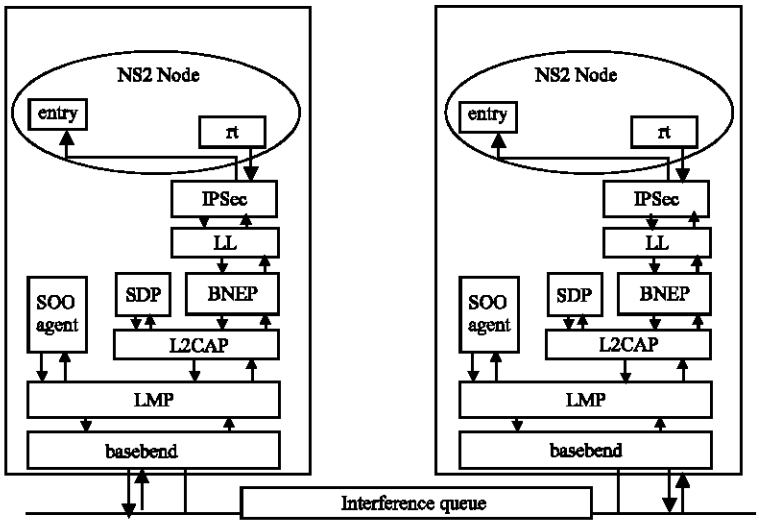

Fig. 4: IPSec architecture in Bluetooth

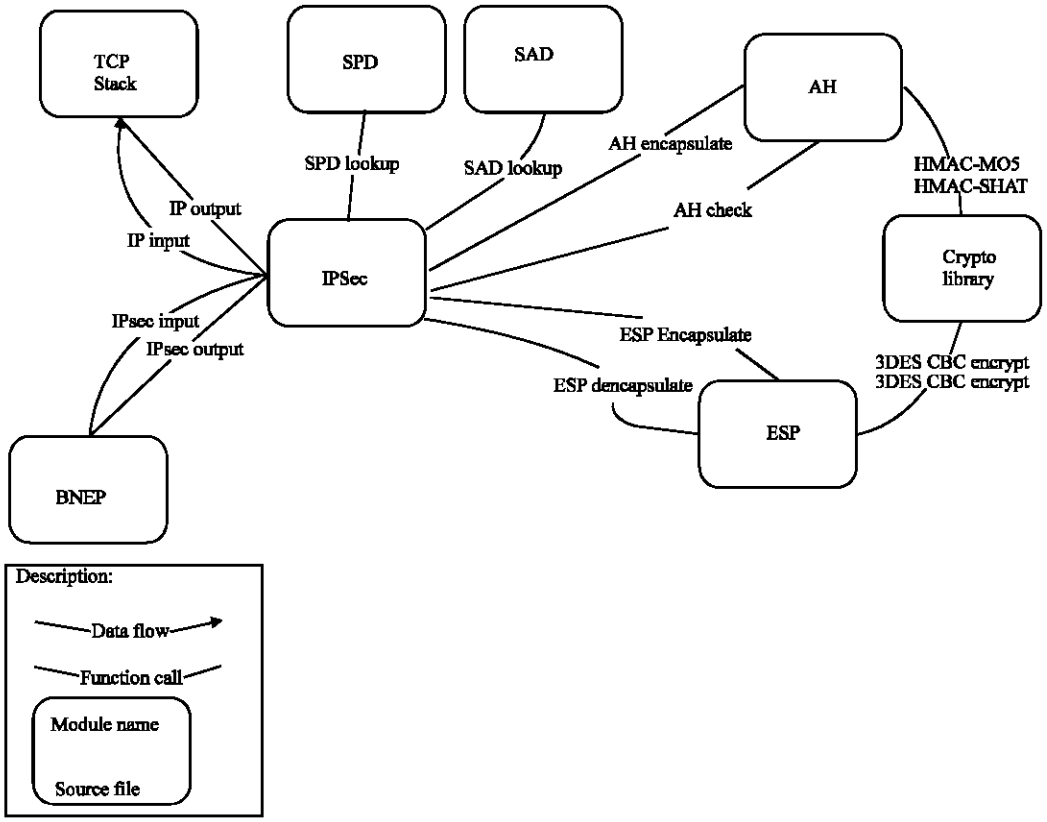

Fig. 5: IPSec system with dependencies

especially those with large installed bases like PPP, BT gives up some efficiency and flexibility that it could have had if more BT specific protocols had been defined. This is a second tradeoff in BT networking design. With future definitions of additional profiles (and maybe additional protocols), such constraints on BT usage might be relaxed.

QoS support: BT provides some support for bandwidth allocation and latency control. FEC (Forward Error Correction) and retransmission mechanisms ensure low error rates at the expense of more overhead and the protocols ensure in-sequence delivery of packets so that reliable, orderly delivery of packets is expected. Link delay and delay jitter are also not expected to be large, once a link has been established.

Mobility support: Partial support for handoff is provided by BT. the fact that a device can be a member of multiple piconets at the same time means that it could in theory perform soft handoffs. However, it is up to the applications to include other necessary features, e.g., the bridging, switching/routing and buffering mechanisms in the backbone.

The IPSec Module is the central part (Fig. 5), which does the whole standard conform processing of the incoming and outgoing IP traffic. It uses a set of data bases (SPD and SAD) to determine the flow of the IP packets. The main processing is then done in the AH and ESP module. A small cryptographic library contains all the functionality used to encrypt, decrypt or to authenticate the packets.

Since there are many security protocols in terms of algorithms in IPSec, we decided on:

- HMAC-MD5 and HMAC-SHA1 to provide origin authentication and integrity for IP packets. MD5 should be preferred because its performance is much better than that of SHA1(Chaudhry *et al*., 2002).
- In ESP implementation we support both encryption and authentication. Encryption is done by the widely used 3DES algorithm, which is applied in CBC mode. Pure DES is also implemented. For authentication we use HASH-MAC MD5.

We needed to identify the different modules out of the IPSec architecture so that we were able to characterize the following attributes of the modules:

- Priority
- Dependencies
- Performance sensibility

An important part of our work was to find a suitable IP-stack and Bluetooth stack that is able to carry our IPSec implementation. The Network Simulator NS-2 TCP/IP Stack has all the desired features: modular design, active community and free BSD-style license. As well as UCBT which has all the desired features needed for Bluetooth stack.

From Fig. 5, any inbound data is forwarded to IPSec input function. Depending on the protocol field in the packet header, the entire packet is forwarded to the IP protocol stack. If the packet could be identified as belonging to the suit of IPSec protocols, it is transferred to the IPSec library. Pure IPSec specific processing, such as applying ESP de-/encapsulation or AH de-/encapsulation is done within the IPSec library.

After these steps, the original IP packet is rebuilt by applying new offsets and packet length to the pbuf structure. Then the clear-text packet is passed up to the TCP/IP stack in NS-2.

For outbound packets, all IP based protocols forward their data to IPSec output function. Here the decision is made whether the packet needs IPSec processing or not. Depending on the appropriate Security Association, AH or ESP functionality will encapsulate the packet. After these steps, the packet is forwarded to the BNEP Class of Bluetooth Stack and sent over to the receiver.

The Security Policy Database (SPD) can be accessed from the IPSec module as shown in Fig. 5. This database contains all rules required to decide how to handle packets, which have security associations but also how to handle non-IP traffic. There are several possibilities: any non-IPSec packet can be forwarded to the default protocol handler (in order for connections from non-IPSec nodes are accepted) or any non-IPSec packet can be dropped immediately without wasting CPU time on further analysis.

BASIC CONCEPT OF SECURITY ASSOCIATION

IPSec needs the Security Policy Database and the Security Association Database to process packets correctly.

The SPD defines the packets, to which IPSec needs to be applied. To guarantee that each packet is processed the right way, each IP packet leaving or entering the system must be checked against the SPD. We call this action the SPD lookup. This lookup does nothing except compare the selectors from the database with the ones from the packet. The SPD lookup delivers back the following results:

- BYPASS: This packet is forwarded directly to Bluetooth layers without applying IPSec.
- DISCARD: This packet is discarded, it will be dropped.
- APPLY: This packet requires IPSec processing

If the result of a SPD lookup is BYPASS, the unmodified packet is forwarded to the Bluetooth layers. This is particularly useful if certain protocols such as ICMP should not be protected by IPSec or communication with non-IPSec hosts must be concurrently possible.

The DISCARD rule is returned when the intention is not to process this packet. If this is the case, the packet will be dropped. This means that we simply delete the packet instead of passing it to Bluetooth layers. It is possible to use this feature to build a primitive firewall.

IPSec processing is only needed if the result of the SPD lookup is APPLY. Whenever a packet matches an SPD entry whose policy says APPLY, then there must also be an SA that describes exactly how the packet has to be processed.

A successful SPD lookup provides us with a pointer to the SP over which we can access the SA using a pointer stored in the SP structure. The packet can be processed only after Security Association parameters are successfully negotiated.

When a packet leaves TCP/IP stack, the very first step is an SPD lookup, a determination of how the packet must be processed. When the policy says APPLY, the IPSec process continues. Otherwise the function passes the packet to the Bluetooth stack or returns to the TCP/IP stack without doing anything. After the new IPSec packet has been built, it must be sent out on the Bluetooth stack as shown in Fig. 6.

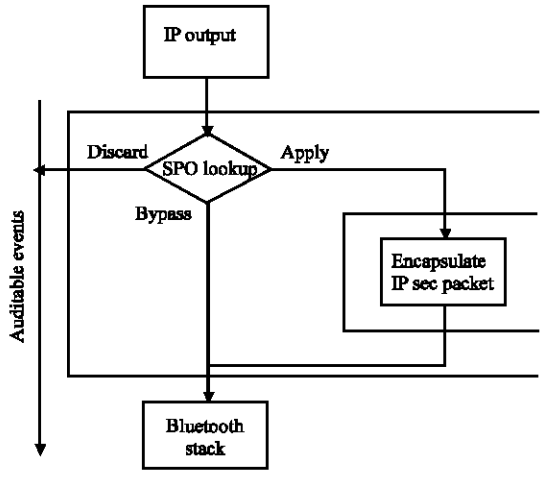

Fig. 6: Outbound processing

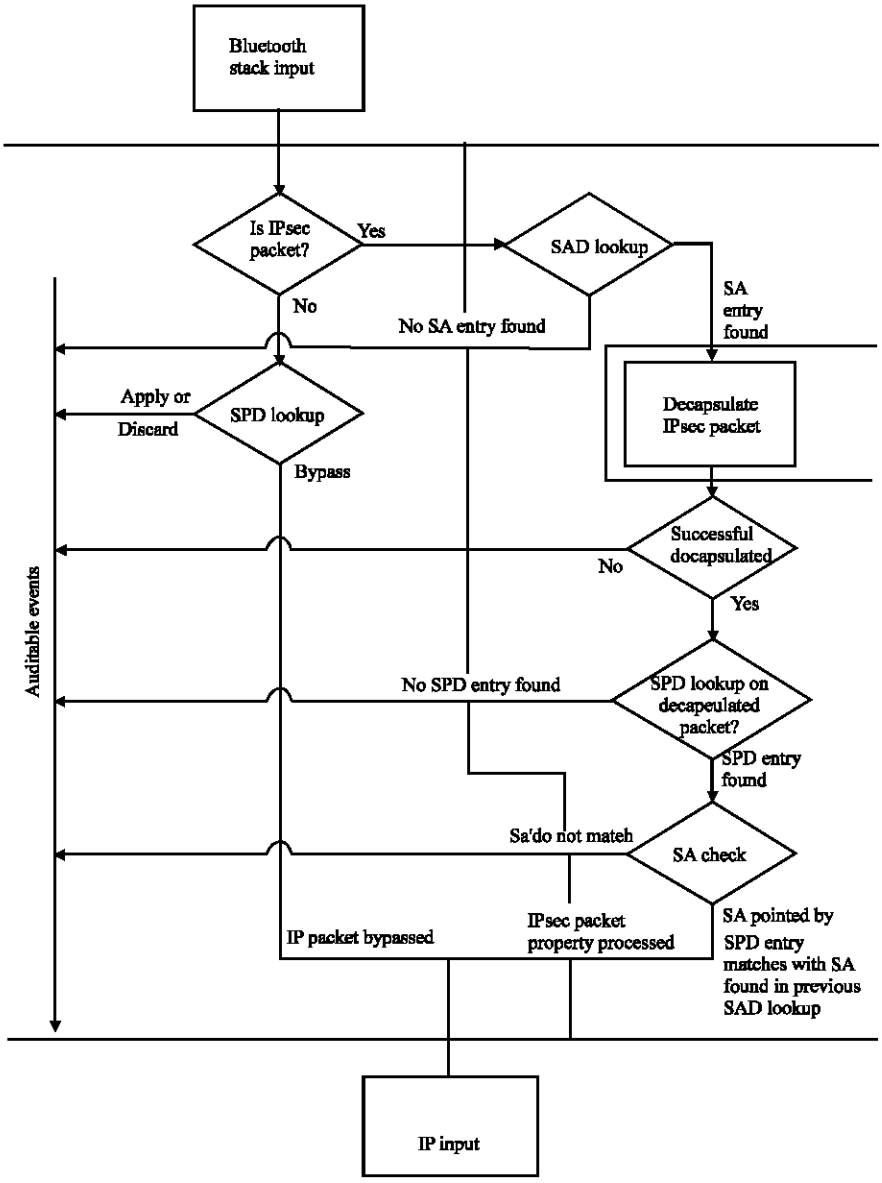

Fig. 7: Inbound processing

From Fig. 7, inbound processing is somewhat different because an incoming IPSec packet already has an SPI, which allows a direct lookup in the SAD table. The reason for using the SPI is straightforward. The incoming IPSec packet may be encrypted and so the SPD lookup, which must be performed on the inner packet data, cannot be performed. The SAD lookup would directly give back an SA if one was found. If no SA is found, then the packet must be discarded.

With the valid SA we are now able to process the packet properly. In inbound processing this corresponds to decapsulation in ESP or integrity checking in AH. After the IPSec packet has been decapsulated, it can be passed on to the TCP/IP.

AH PROCESSING

Authentication is done by the well-known HASH-MAC4 MD5 and HASH-MAC SHA1 algorithms. These are the algorithms requested by the standard. MD5 should be preferred because its performance is much better than that of SHA1.

If one considers that ESP also supports integrity and authentication, one may think that there is no need for AH. This is not true because the authentication and integrity check of AH is a bit more sophisticated. Authentication in AH covers more fields of the packet than ESP does.

AH processing can be split up into inbound and outbound processing.

AH inbound processing itself described step-by-step

- In order to check the integrity and the authentication of the packet, the ICV must be calculated. The ICV calculation in AH also covers the outer IP header. In this header there are so-called mutable fields, which change their value while they are sent across the network. Those fields (Type of Service, Offset, TTL and checksum) must first be set to zero. The ICV fields in the AH header must be backed up and zeroed, so that later comparing remains possible. It becomes clear that AH authentication also covers the source and destination address of the outer IP packet.
- The packet is now ready to be verified and the integrity check value can be calculated over the whole packet. The SA determines the appropriate algorithm and key.
- The calculated ICV can be compared with the one saved in the first step. Processing continues only if the calculated ICV matches the original one.
- The authentication of the packet is now verified and the anti-replay check can be performed. If it is successful, the sequence number (stored in the SA) is incremented. Finally, the offset and packet length are passed back.

AH outbound processing described step-by-step:

- First of all a new AH header is placed in front of the IP packet, leaving a gap between the inner IP header and the AH header. This gap is later used to place the ICV. The AH header fields: next header, length, SPI and sequence number are added.
- After the outer IP header has been constructed, only the source and destination address, version, header length and total length are set. The other fields are set to zero as a preparation for the ICV calculation. Padding is not required because the packet is already aligned.
- The integrity check value can now be calculated and placed into the gab between AH header and inner IP header.
- After the ICV has been calculated, the zeroed fields are now filled with the appropriate values.
- Finally, the offset and the packet length are passed back.

ESP PROCESSING

In our ESP implementation we support both encryption and authentication. Encryption is done by the widely used 3DES algorithm, which is applied in CBC mode. Pure DES is also implemented. For authentication we use HASH-MAC MD5 and HASH-MAC SHA1.

ESP processing can be split up into inbound and outbound processing.

ESP inbound it self described step-by-step:

- A check in the SA structure indicates whether authentication needs to be checked or not. If an authentication algorithm is specified within the SA, the ICV must be calculated and compared with the one stored at the end of the ESP packet. The ICV is calculated over the whole ESP header, IV and encrypted payload. Processing continues only when the packets ICV matches our recalculated one.
- In the next step we have to decrypt the packet. The decryption algorithm and the secret key can be accessed over the SA. Because the packet was encrypted in CBC-mode, the IV must be copied out of the ESP packet. The IV is stored between ESP header and encrypted payload. The decryption happens in-place, so no copying must be done.
- Before everything is done the sequence number counter in the SA is incremented and optionally the same is done with the anti-replay window. To let the caller of the ESP function know about the location and the size of the extracted IP packet, the offset and the packet length are giving back.

ESP outbound processing itself described step-by-step:

- The first step of encapsulation is to test whether the decremented TTL field of the IP header reaches zero. If this is the case, the packet must be discarded in order to prevent endless straying of packets.
- Then we have to calculate how much padding must be added to fulfill the requirements of the encryption algorithm. The right amount of padding bytes is added at the end of the payload. The fields: padding length and next header are appended right after the padding.
- Encryption is performed according to the settings in the SA. After encryption, the used IV is copied in front of the encrypted payload.
- ESP header is added in front of the IV. Inserted are a incremented sequence number and the SPI taken out of the SA.
- The SA must be checked to see if authentication is enabled. If this is the case, then the ICV must be calculated according the SA's settings. The ICV, which is calculated the ESP header, the IV and the encrypted payload, is copied at the end of the payload.

- Outer IP header can be constructed using the tunnels source and destination address given as input arguments to the function. The TOS field is copied from the inner IP header. Finally, the offset and the length are passed back, so that the caller can update its data structure, where the packet is stored.

RESULTS AND PERFORMANCE ANALYSIS

IPSec over Bluetooth scenario was simulated with a constant FTP source on top of TCP with a packet size of 2000 bytes between two nodes (node 0 and node 1).

Table 1 show the Simulations of the system design described in section 5 using different algorithms using the hand-off rate of 60 seconds. HMAC- MD5 has shown to have the highest No of Bytes sent where 3DES HMAC-MD5 has lowest.

The difference in simulations' time varies from one algorithm to another due to the delay required for the authentication or encryption procedure. As delay consumes more time, it affects the No. of packets generated. As shown in Table 1, HMAC-MD5 algorithm has the lowest delay but the highest No. of packets generated, where 3DES HMAC-MD5 algorithm has the highest delay but the lowest No. of packets generated.

It's also noticed that the average packet size is not fixed due to different types of packets (TCP, Authenticated packets, Encrypted Packets, ACK, etc...,) sent between nodes where each has a different size.

Authenticated packets (HMAC-MD5 and HMAC SHA1) almost have the same average packet size while 3DES HMAC-MD5 has higher average packet size in encryption (ESP).

In order to study the Performance of IPSec over Bluetooth, we measured the cumulative sum of packets in each of the cases, as well as the throughput and end 2 end delay imposed by the security protocols IPSec.

Figure 8, depicts the decrement of the TCP packets sequence in different scenarios (No IPSec, MD5, SHA1, DES-MD5 and 3DES-MD5). The reason behind this decrease in No. of packets is the Authentication and Encryption Procedure which includes SPD lookup when the result is DISCARD. This means that we simply delete the packet instead of passing it to Bluetooth layers, so the packet is lost thus more delay is consumed and wasted The Fig. 8a and b scenarios showed that number of packets is decreasing in authentication and encryption compared to Bluetooth with no IPSec. As it is seen HMAC-MD5 performs better than HMAC-SHA1 while sending and receiving packets. As well as DES-MD5 performs better than 3DES-MD5.

Plain Bluetooth packets and HMAC-MD5 share almost the same performance at trade-off rate 7, but later on HMAC-MD5 start losing packets and decrements more.

On the other side, SHA1, DES-MD5 and 3DES-MD5 share almost the same performance at trade-off rate 12 where SHA1 starts catching up with MD5, while DES-MD5 and 3DES-MD5 remain sharing it till trade-off rate 18.

Throughput is defined as the percentage of packets that experience a Bit Error Probability (BEP) that is less than a maximum allowable BEP, BEPth. The BEPth is set to a value of 0.1% (or 10^{-3}), i.e., a packet is considered reliable as long as its BEP is not greater than BEPth. The quality of the link varies from one BT unit to the next within the scatternet. It is highly dependent on the relative spacing of the BT devices within the scatternet. Hence, the percentage of reliable packets has to be determined by collecting the statistic over many realizations of the scatternet. The conservative measure of 10th percentile is used to represent the throughput performance of the BT network.

The throughput results are shown in Fig. 9. There is a significant difference between the simulated scenarios. As we can see, the throughput in HMAC-MD5 and HMAC-SHA1 is not the same due to a better performance from HMAC-MD5. In respect to the encryption, DES-MD5 has a better throughput than 3DES-MD5.

In sending packets, plain Bluetooth packets throughput remain at a rate of 78, HMAC-MD5 throughput is at around 50 while HMAC-SHA1 at a rate of 40. For encryption algorithms, throughput rate is 15 for DES-MD5 and 8 for 3DES-MD5.

These rates decrease while receiving packets; they become around 37 for plain Bluetooth packets, 28 for HMAC-MD5, 20 for HMAC-SHA1, 6 for DES-MD5 and 2 for 3DES-MD5.

Table 1: Retrieved initial results

Parameters	No. IPSec	HMAC MD5	HMAC SHA1	DES HMAC- MD5	3DES HMAC- MD5
Simulation start time	1.30513601	1.4471	1.46	1.5374	1.6665
Simulation end time	60.08551301	60.073	59.6643	59.9108	60.0448
Simulation length (Sec)	58.7880377	58.62592863	58.20427423	58.37336781	58.37830156
No of generated packets	4500	3501	24439	658	237
No of lost packets	2260	1753	1222	330	119
No of received packets	2240	1648	1217	328	118
Avg. packet size	1051.7595	1066.1472	1066.2114	1074.5023	1069.6913
No of sent bytes	4698000	3700136	2578664	701480	251668

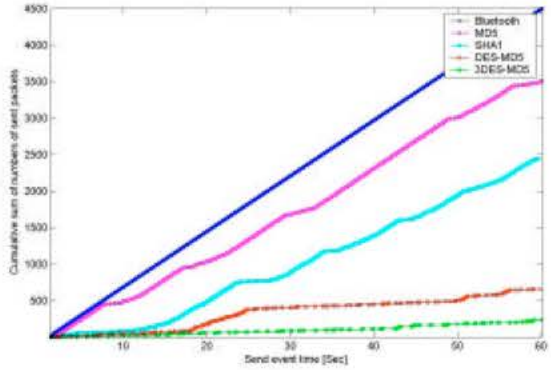

Fig. 8a: Performance of Cumulative sum of numbers of sent packets

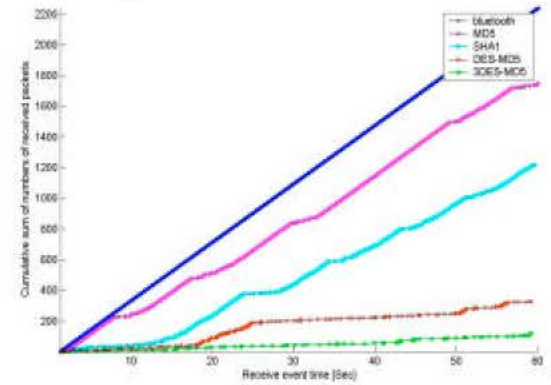

Fig. 8b: Performance of Cumulative sum of numbers of received packets

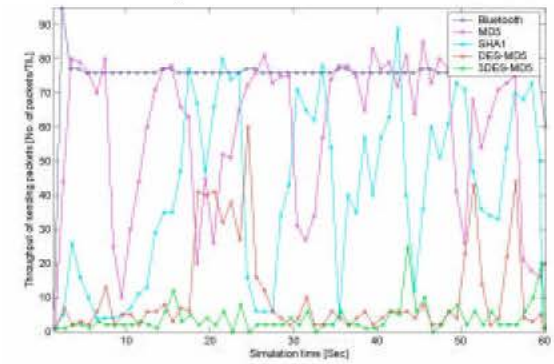

Fig. 9 (a): Throughput of sending packets

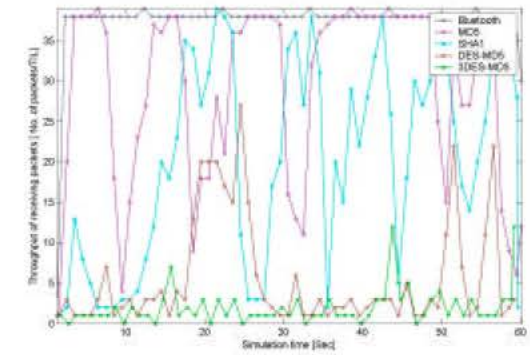

Fig. 9b: Throughput of receiving packets

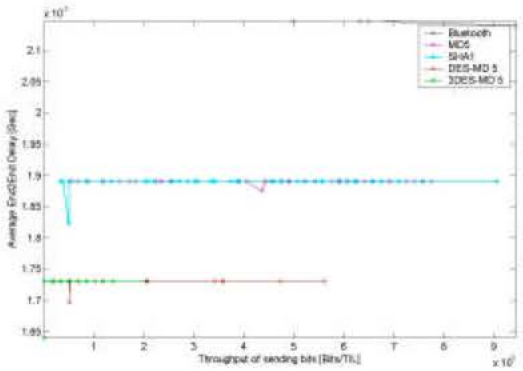

Fig. 10a: Throughputs of senidng bits vs. average simulation end 2 end delay

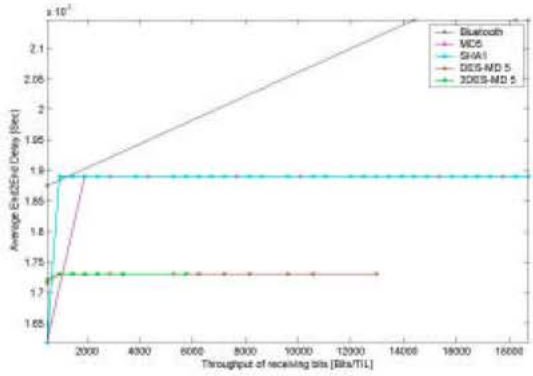

Fig. 10b: Throughputs of receiving bits vs. average simulation end 2 end delay

Contrary to our expectations, the throughput in Bluetooth environment is driven by the effect of:

- The successive sending of constant packets, whether its ACK packet where throughputs increases or encrypted and authenticated packets in which throughput decreases.
- The erratic behavior of delay imposed by the encryption and decryption of the data.
- The erratic behavior of the Bluetooth wireless link, Wireless links are characterized by higher bit error rates and this causes inefficiencies in the operation of TCP. Essentially, any perceived packet loss (occurring because of error or buffer overflow) is construed by a TCP sender as occurring due to buffer overflow., The response of TCP to all such events is to invoke its congestion control procedures, resulting in unnecessary window reduction, which causes a drop in the TCP throughput. Note, though, that some of the packet losses occur due to corrupted packets being dropped by the link layer and invoking the congestion avoidance procedures when these events occur is not desirable.

Figure 10 show a comparison of Throughputs vs. average simulation end 2 end delay. There is a big and noticed difference of end 2 end delays while sending the bits and rise in the throughput. While in receiving bits, both factors also affect the bits transmission.

The main difference noticed raises at throughput of sending and receiving packets, while end 2 end delay almost equal in case of Authentication algorithms, it's also the same as for Encryption algorithms.

CONCLUSION

This study proposed a new Bluetooth security scheme, which allows ad-hoc (PAN) based on Bluetooth technology to communicate with other devices in full secure channel includes authentication and encryption, unlike for the present schemes with weak security (E0 and E1).

In addition, as shown in Fig. 11, the throughput in sending packets is reduced by almost 35-50% for authentication and 80-90% for encryption compared with the cases where IPSec was not used. While in receiving packets, throughput is reduced by almost 30-45% for authentication and 80-95% for Encryption.

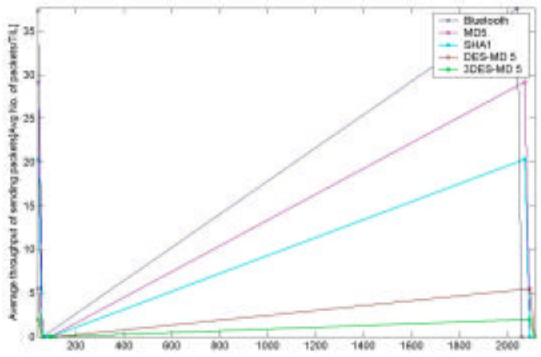

Fig. 11a: Packet size vs. average throughput of sending packets

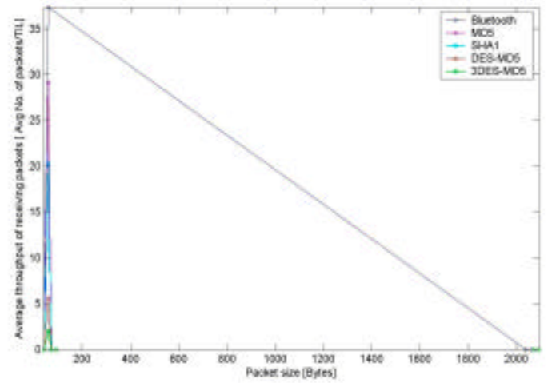

Fig. 11b: Packet size vs. average throughput of receiving packets

The throughput in Bluetooth environment is driven by the effect of the successive sending of constant packets, the erratic behavior of delay imposed by the encryption and decryption of the data, the erratic behavior of the Bluetooth wireless link and Wireless links are characterized by higher bit error rates and this causes inefficiencies in the operation of TCP.

We have not addressed the issues related to Bluetooth radio layer and polling algorithm at the baseband, where problems such as noise, interference and packet loss, may have a significant impact on performance (Misic and Jelena, 2003).

Over all and according to the results in the previous section, it has been proved that HMAC-MD5 performance better than HMAC-SHA1 for authentication and DES/ HMAC-MD5 reliable than 3DES/ HMAC-MD5 for Encryption.

REFERENCES

Anonymous, 1980. The Ethernet-A Local Area Network, September 1980. Version 1.0 Digital Equipment Corporation, Intel Corporation, Xerox Corporation.

Bluetooth, [referred 2001-02-11]. The Bluetooth Specification, v.1.1B. < http: //www. bluetooth. com/developer/ specification/ specification.asp >

BNEP_specification_rev10RC3 [referred 2002-12-17]. <www. grc. upv. es/localdocs /blue/ BNEP_specification_ rev10RC3. pdf >

Chaudhry, G., D. Medhi and J. Qaddour, 2002. Performance Analysis of IPSec Protocol: Encryption and Authentication, IEEE Communications Conference (ICC 2002).

Internet Engineering Task Force, 1996. A Standard for the Transmission of IP Datagram's over Ethernet Networks, RFC 894.

Internet Engineering Task Force, 1998. Classical IP and ARP over ATM, RFC 2225.

Internet Engineering Task Force, 1999. IPv4 over IEEE 1394, RFC2734. <http:// www.iana. org/assignments/ ethernet-numbers>.

Kelsey, J., B. Schneier and D. Wagner, 1999. Key Schedule Weakness in SAFER+, Second AES Candidate Conference.

Lu, Y. and S. Vaudenay. Faster Correlation Attack on Bluetooth Keystream Generator E0, CRYPTO 2004, pp: 407-425.

Misic, B. Vojislav and Jelena, 2003. Polling and Bridge Scheduling Algorithms in Bluetooth. <http:// www.cs.umanitoba.ca/~vmisic/pubs/tr0304.pdf>

Nomination of SAFER+ as Candidate Algorithm for the Advanced Encryption Standard (AES), 1998. Submission document from Cylink Corporation to NIST.

RFC 2401, 1998. Security Architecture for the Internet Protocol, available at<http: //www. ietf. org/rfc /rfc2401. txt>.

Saarinen, M.J., 2000. A Software Implementation of the BlueTooth Encryption. Algorithm E0, < http://www. jyu. fi/~mjos/e0.c >.

The network simulator NS-2. < http://nsnam.isi. edu/nsnam/index. php/User_Information>

UCBT - Bluetooth Extension for NS2 at the University of Cincinnati < http://www.ececs.uc.edu/~cdmc/ucbt>