Q-ESP: a QoS-compliant Security Protocol to enrich IPSec Framework

Mahmoud Mostafa, Anas Abou El Kalam, Christian Fraboul

Q-ESP: a QoS-compliant Security Protocol to enrich IPSec Framework

Mahmoud MOSTAFA, Anas ABOU EL KALAM, Christian FRABOUL
Université de Toulouse, INPT-ENSEEIHT, IRIT-CNRS, 2 rue Camichel Toulouse –France.
{firstname.lastname}@enseeiht.fr

***Abstract*—IPSec is a protocol that allows to make secure connections between branch offices and allows secure VPN accesses. However, the efforts to improve IPSec are still under way; one aspect of this improvement is to take Quality of Service (QoS) requirements into account. QoS is the ability of the network to provide a service at an assured service level while optimizing the global usage of network resources. The QoS level that a flow receives depends on a six-bit identifier in the IP header; the so-called Differentiated Services code point (DSCP). Basically, Multi-Field classifiers classify a packet by inspecting IP/TCP headers, to decide how the packet should be processed. The current IPSec standard does hardly offer any guidance to do this, because the existing IPSec ESP security protocol hides much of this information in its encrypted payloads, preventing network control devices such as routers and switches from utilizing this information in performing classification appropriately. To solve this problem, we propose a QoS-friendly Encapsulated Security Payload (Q-ESP) as a new IPSec security protocol that provides both security and QoS supports. We also present our NetBSD kernel-based implementation as well as our evaluation results of Q-ESP.**

***Index Terms*— ESP, IPSec, Performance evaluation, QoS, Security protocols.**

I.INTRODUCTION

In today's business environments, users demand seamless connectivity and stable access to servers and networks wherever they are: hotels, airports, remote offices, etc. While these functionalities are useful for business, they work only if we can minimize the security risks of transferring sensitive data across the Internet. In order to achieve this goal, there are various security mechanisms for network environment; the most popular is Security Architecture for IP (IPSec) [1]. IPSec is designed to provide inter-operable cryptographically-based security for lPv4 and lPv6. IPSec operates at the IP layer, making it transparent to applications and users.

Unfortunately, security does not come for free and, in general, protection mechanisms require more processing time and causes traffic delay. Real-time applications such as video conferencing, VoIP, and real-time video need special processing to achieve their goals and to overcome the delay introduced by adding security mechanisms. Quality of service (QoS) has been emerged to solve a part of this problem by providing priority treatment to real time traffic. In the QoS domain, the Class of Service concept divides the network traffic into different classes and provides a class-dependent service to each packet. To classify packets, each packet is assigned a priority value. The latter is stored in the "Type of Service" (ToS) field in the IP v4 header (also called Traffic Class in IP v6).

However, allowing the sending device to classify traffic or to set traffic priorities may be subject to threats, as the sender may classify his traffic in a way that gives him upper priorities. This is clearly the disadvantage of what is called *passive admission control*. Conversely, service providers perform *active admission control* by allowing edge routers (neither users nor the sending devices) to inspect the incoming traffic and classify it. The component in charge of this task is called *Multi-Field classifier* (*MFC*) [2].

The inspected fields belong to different network layer headers [3]:

- at Transport Layer Protocol Header: in order to identify the applications running over TCP or UDP, the source and destination port numbers are inspected.
- at Network Layer Protocol Header: *MFC* inspects the source host IP address, the destination host IP address and the protocol identifier (that is used to identify the transport-layer protocol in use).

The previously mentioned five fields of the transport and network protocols headers are used to define the traffic flow [4]. They are collectively called "*five-tuple*". Unfortunately, even if these fields are required for QoS processing, some of them are hidden (encrypted) when using IPSec ESP [5] security protocol. IPSec ESP protocol encrypts the transport layer header, and thus hides the source and destination port numbers as well as the protocol identifier.

To solve this problem, we introduce the Q-ESP "*QoS friendly Encapsulated Security Payload*" protocol that enforces both QoS and security requirements. The major aim of our Q-ESP is to construct packets that are QoS controllable according to active admission control while providing the same security services ensured by IPSec ESP and AH [6].

The remainder of this paper is organized as follows: Section II presents our Q-ESP protocol packet structure. Then, Section III gives the details of Q-ESP processing. Section IV focuses on Q-ESP implementation. Afterward, Section V, provides our Q-ESP analytical evaluation. In Section VI, we present the performances evaluation experiments and results. Finally, Section VII draws up conclusions and future works.

II. Q-ESP PROTOCOL PACKET STRUCTURE

Figures 1 and 2 depicts the structures of Q-ESP protocol packets in IP versions 4 and 6 respectively. Basically, Q-ESP packets contain the header, the payload, the trailer, and the authentication data area.

The next sub-sections give details of these fields.

A.The Q-ESP header

As mentioned above, to satisfy QoS requirements, we copy the first two fields (source and destination ports) of the upper layer transfer protocols and place them as such (without encryption) in the beginning of the Q-ESP header. Then, we add the TLP (Transport Layer Protocol number) field. In this respect, the Q-ESP header includes the following six fields:

1) *Source port*: this 16-bit field contains the first field of the upper layer transport protocol (TCP/UDP) source port number.
2) *Destination port*: this 16-bit field contains the second field of the upper layer transport protocol (TCP/UDP) destination Port number.
3) *TLP*: this 8-bit field contains the Transport Layer Protocol number (6 for TCP and 17 for UDP).
4) *Reserved*: is a 24-bit field that is currently not used and must be set to zero.
5) *Security Parameters Index (SPI):* is a 32-bit fixed length identifier that uniquely identifies the Security Association (SA). For each protected connection, the SA contains information such as the used secret keys, hash algorithm (e.g., SHA-1) [7], encryption algorithm (e.g., AES) [8]. The SPI helps the recipient to identify which SA will be used to process the packet.
6) *Sequence Number*: is a 32-bit fixed length field. It is a monotonically increasing ID that is used to prevent replay attacks. This value is authenticated, so that malicious or accidental modifications could be detected.

B.The Q-ESP payload

The payload encrypts the upper layer transport protocol and its payload data in transport mode, while in tunnel mode it encrypts the entire original IP packet.

C.The Q-ESP trailer

The Q-ESP Trailer contains the following fields:

1) *Padding:* allow block-oriented encryption algorithms area in multiples of their block size.
2) *Pad length:* is an 8-bit field that indicates the length of the included pad.
3) *Next Header:* refers backward to the protocol (IP, TCP, UDP, etc.) in the encrypted payload.

D.Authentication data area

It is a variable length area that is used to store the Integrity check value (ICV); the latter is computed by the authentication algorithms (e.g., SHA-1 [7]). The ICV is calculated over the Q-ESP headers and the Payload. More over, in our work, to inherit the capability of IPSec AH, we also apply the authentication algorithm over the source IP address and destination IP address fields of the IP header.

The next section describes, among others, how the ICV is calculated (step 05) and verified (step I4).

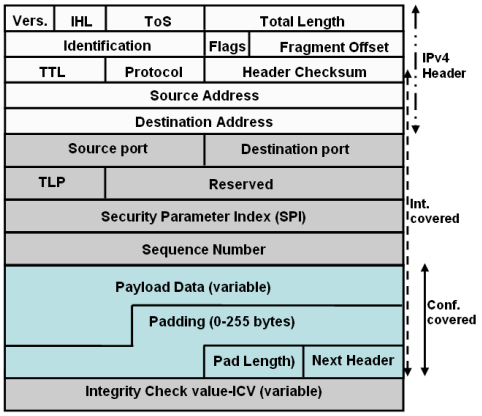

Fig. 1. Q-ESP packet format in Ipv4.

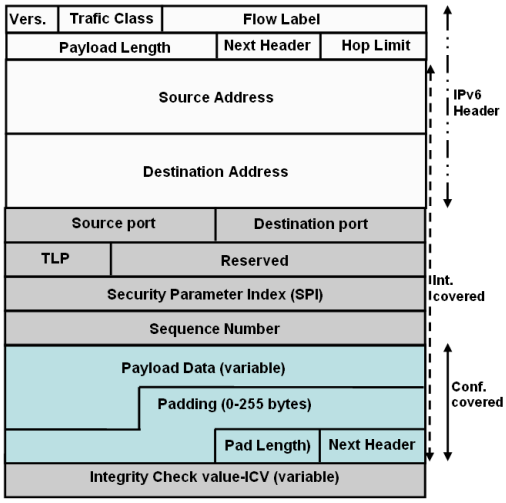

Fig. 2. Q-ESP packet format in Ipv6.

III.Q-ESP PROCESSING

A.Outbound processing

Step O1: Searching SA

Upon reception of an outbound IP packet, refer to the Security Policy Database (SPD) to determine if Q-ESP is applied to this packet; if yes, search Security Association Database (SAD) to extract SA to be used. If it exists, proceed to the next step; if not, (no SA exists) instruct IKE to build it. From this step, we obtain an SA that contains an SPI that uniquely identifies the SA, an initialized sequence number, the algorithms to be used for encryption and authentication, their secret keys as well as the mode of operation (either tunnel or transport mode).

Step O2: Constructing the IP header

The mode of operation affects this step; for transport mode, the old IP header will be used to send the packet, while for tunnel mode we construct a new IP header to tunnel the old IP packet. To construct the new outer IP header, the new source

and destination IP addresses of the gateways will be placed in the outer IP header. The remaining fields will be copied from the old IP header to their corresponding fields in the new outer IP header. After that, the time to live TTL field will be decremented.

Step O3: Constructing the Q-ESP header

To construct Q-ESP header, we will copy the first two fields (source and destination ports) of the upper layer header protocol (TCP/UDP) to the beginning of Q-ESP header. Then, we will put the protocol number of the upper layer transmission protocol in the TLP field. Next we will set the value of the reserved field to zero. After that, we will place the security parameter index (SPI) obtained from the SA in its field (to tell the receiver how to react with this packet); and finally, we will increment the sequence number and place it at the last field of the header.

Step O4: Encrypting the Q-ESP payload

To encrypt the Q-ESP payload: padding is added, its length is recorded, and the whole Q-ESP payload and its trailer are encrypted according to equation (1).

$$M_E = E (M_P \| T_r , K_E) \quad (1)$$

Where, M_E is the encrypted message, E is the encryption function, M_P is the Q-ESP payload, T_r is the Q-ESP trailer and K_E is the encryption / decryption key.

It is important here to note that, Q-ESP payload contents differ according to the mode of operation. For transport mode, the whole upper layer protocol is placed in the Q-ESP payload, while in tunnel mode, the original IP header is placed before the upper layer protocol in the Q-ESP payload.

Step O5: Computing the authentication value

Use the specified authentication algorithm to compute ICV according to equation (2).

$$ICV = H(\text{Src IP} \| \text{Dst IP} \| M_H \| P, K_A) \quad (2)$$

Where, H is the keyed-authentication algorithm, and the "Src IP" and "Dst IP" are the source and the destination IP addresses fields of the outer IP header, M_H is the Q-ESP header, P is the Q-ESP payload, and K_A is the authentication key.

Step O6: Finalizing

In this step the Q-ESP packet construction is completed and the IP header checksum is to be computed.

B.Inbound processing

In this section we describe how the receiver will reconstruct the original packet from the Q-ESP.

Step I1: Searching SA

Upon reception of an inbound Q-ESP packet, search SAD by the SPI to retrieve SA to be used; if it exists, proceed to the next step, if no SA exists for this SPI, drop the packet and instruct IKE to build a new SA.

Step I2: Checking sequence number

In Q-ESP protocol, this step is mandatory to prevent replay attacks. If the sequence number of the packet is valid (i.e., it is not a duplicate and is not to the right of the sequence number window contained in the SA), proceed to the next step, otherwise the packet is dropped.

Step I3: Verifying the authentication value

Use the specified authentication algorithm to re-compute ICV (using equation (2)) for the source, and destination IP addresses fields of the external IP header, the Q-ESP header and its payload. Then, the result is compared with the value stored in the Authentication data area; if they are equal, proceed to the next step, if not, drop the packet.

Step I4: Decrypting Q-ESP payload

Use equation (3) to decrypt payload.

$$M_D = D (M_E , K_E) \quad (3)$$

Where M_D is the decrypted results (Q-ESP payload and its trailer) and D is the decryption function.

Step I5: Extracting the encapsulated protocol from Q-ESP

This step is affected by the mode of operation. For the transport mode, extract the upper layer protocol and its payload data from the Q-ESP payload. For tunnel mode, extract the inner-IP packet from the Q-ESP payload.

IV.Q-ESP IMPLEMENTATION

We modified the IPSec kernel implementation of NetBSD version 5 [9] to implement Q-ESP protocol. Actually, we have chosen NetBSD to benefit from its Fast IPSec implementation and its rich cryptographic framework, which is the highest performance publicly available IPsec implementation [10].

Our implementation also provides a *Userland* patches for the *Setkey* tool (for a manual configuration of IPSec). In addition to our patches, we develop a *Wireshark plug-in* [11] in order to simplify the debugging and validation of Q-ESP protocol.

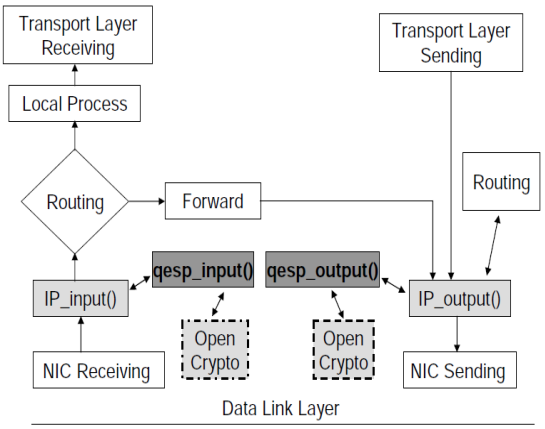

Figure 3. Q-ESP implementation.

A.Implementation of Outgoing Packets' Processing

As shown in Fig. 3, when an IP packet is sent, the *ip_output()* routine is triggered, then, a lookup in Security Policy Database (SPD) is performed by *ipsec_output()* routine. If we should use Q-ESP, the *qesp_output()* routine constructs and inserts the Q-ESP header into the packet. Then, it applies the concerned cryptographic processing using *Opencypto* cryptographic interface. After that, *Opencrypto* calls the

qesp_output_cb() routine and delivers the protected packet back to the *ip_output()* which in turn uses the Network Interface Card (NIC) to send the packet.

B.Implementation of Incoming Packets' Processing

Incoming packets -with the protocol identifier set to Q-ESP protocol number- are delivered to *qesp_input()* by *ip_input()* and *ipsec_input()* routines. The *qesp_input()* lookup the SPI of the Q-ESP packet in SAD. Then, the packet is forwarded to the appropriate *Opencypto* interface routine to perform the necessary cryptographic transforms. After that, the *qesp_input_cb()* routine extracts the payload and sends the IP packet back to *ip_input()* which delivers the payload to the specified transport protocol.

V. ANALYTIC EVALUATION

In this section, we give our analytic comparison of Q-ESP versus IPSec ESP and AH. The comparison covers both QoS and security aspects. Let's first briefly describe the structure and functionality of ESP and AH.

A.Authentication Header (AH)

The AH [6] protocol provides data origin authentication, anti-reply integrity, and connectionless integrity; note that the anti-reply service is optional as it may not be enforced at the receiver's side. Fig. 4 represents AH header structure.

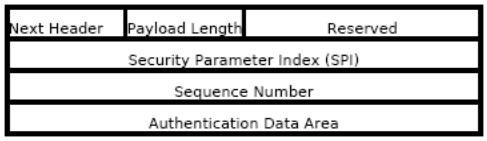

Fig. 4. The AH format.

Basically, the sender uses an authentication algorithm to compute the integrity check value (ICV) for the entire IP packet, and stores the result in the authentication data area.

B.Encapsulated Security Payload (ESP)

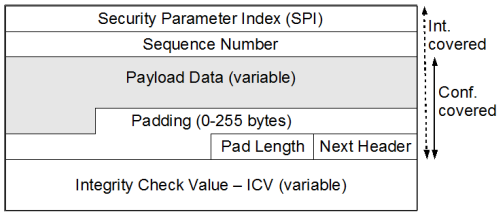

Fig. 5. The ESP format.

The Encapsulated Security Payload (ESP) [5] mainly provides data confidentiality (through encryption). Fig. 5 shows the structure of the ESP header. Unlike AH, ESP surrounds the payload that it protects. First ESP prepares the payload for the encryption algorithm by adding padding, recording its length and copying the value of the protocol field from the IP header to the Next header field in the ESP trailer. After that, the encryption algorithm encrypts the ESP payload and its trailer. As an optional service the integrity check value is computed and placed in the authentication data area. Note that unlike AH, ESP authentication does not cover the outer IP header. Fig. 6 shows ESP in both transport and tunnel modes. The dotted arrows represent the authentication coverage, while solid arrows represent the encryption coverage.

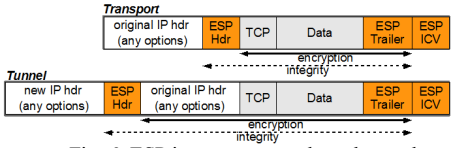

Fig. 6. ESP in transport mode and tunnel mode.

C.Q-ESP versus ESP and AH

In addition to providing the same security services provided by IPSec ESP and AH, Q-ESP supports QoS by providing the necessary and sufficient information to perform active admission control.

Besides that, Q-ESP prevents replay attacks. Note that while the anti-reply function is optional in ESP and AH, it is mandatory in Q-ESP. In addition, while authentication is optional in ESP, both encryption and authentication are mandatory in Q-ESP. Authenticating the packet after encrypting it helps to prevent DoS attacks [12] that form malicious packets from valid IP and ESP/ Q-ESP headers, but with invalid payloads. Moreover, Q-ESP provides data origin authentication (as it covers the source and destination IP addresses of the outer IP-header).

Moreover, the Q-ESP header is small (16 bytes) while the sum of the AH and ESP headers is large (8 for AH + 12 for ESP = 20 bytes). Therefore, the Q-ESP header overhead is smaller than the overhead of the two protocols (ESP and AH) used simultaneously. We think that it is not fear to compare Q-ESP header with ESP or AH header separately, as Q-ESP provides the same security services provided by the two security protocols (AH and ESP used conjointly).

Regarding time complexity, Q-ESP has mainly the same processing steps as ESP and AH; the only two differences:

First at the sender site, Q-ESP copies the two upper layer transport protocol (source and destination port numbers) to its header; the time needed for this process is thus very short and can be neglected.

The second difference is the authentication coverage. ESP authenticates only its header and the protected payload (it does not authenticate the outer-IP header) while AH authenticates the entire packet including its outer-IP Header. Actually, the authentication in Q-ESP protocol covers: its header, the protected payload, and only two fields from the outer-IP header (source and destination IP addresses). It seams that, this difference in the authentication coverage may affect the performances. More precisely, assuming that we use the SHA-1 [7] algorithm for the integrity control, and based on the analysis presented in [13], the time complexity of the SHA-1 authentication is: n * 1110 operations, where n is the number of input blocks given by: n=ceiling (total message size/ (block size =512)). Note that as the authentication algorithms do not accept incomplete blocks; padding are added to the end of the

message to complete block size if needed.

Besides that, while comparing Q-ESP and ESP, it seems interesting to put the following two extreme situations into account: either Q-ESP has the same number of blocks (n) as ESP or, in the worst case; Q-ESP has (n+1) blocks as its authentication coverage includes more fields. In the same way, while comparing Q-ESP and AH, we can consider the same two extreme situations: either AH has the same number of blocks (n) as Q-ESP or AH has (n+1) blocks as its authentication covers the entire outer IP header. The additional block will cost 1110 operations in case of SHA-1 as additional processing overhead.

VI. Experiments for performance Evaluation

In this section, to precisely analyze the performance of the Q-ESP, we conducted two experiments with two different testbeds. The first experiment was conducted to measure the throughput of Q-ESP against ESP in normal situation without QoS priority treatment. The second experiment was performed in situation of network congestion. The goal is to prove that Q-ESP is active admission controllable, can be used to provide priority treatment for time critical traffic, and performs well in the situation of network congestion.

A. The Benchmark Tool

To generate UDP traffic, and thus simulate a real-time application, we choose the Multi-Generator (MGEN) tool [14]. At the sender side, MGEN allows generating constant UDP packets as well as random traffic according to probabilistic laws. At the receiver side, MGEN can also create a listener to receive this traffic. Moreover, MGEN logs can be used to calculate performance statistics.

In addition, if used with the NTP protocol [15], MGEN can record transmission and reception times of individual packets in a synchronized way between all hosts.

Besides that, for generating TCP traffic, we used the Iperf tools [16] thanks to its simplicity and richness.

A.Throughputs Experiment

The goal of this first experiment is to measure throughput using the two protocols IPSec ESP and Q-ESP. To achieve this goal, we built the testbed shown in Fig. 7.

Fig. 7. Testbed for throughput experiments.

The two security gateways (GW-1 and GW-2) have been connected to each other using point-to-point connections. In addition, we constructed a Network-to-Network IP-VPN using ESP and Q-ESP respectively. To measure throughput, we connected one measurement PC to each security gateway (PC1 and PC2). Due to space limitation, hardware specifications have been omitted

The MGEN has been installed in the measurement PCs. MGEN was used to send UDP packets from PC-1 to PC-2 and measure throughput. The gateway-1 handled outbound processing and gateway-2 handled inbound processing duties. In this experiment Q-ESP and ESP respectively was installed in the two security gateways. Manual SA was created to perform encryption using AES and authentication using HMAC-SHA-1. Table I shows the throughput for ESP and Q-ESP using different packet size.

TABLE I: Q-ESP AND ESP THROUGHPUT (KBPS).

Packet size	ESP	Q-ESP
64	51.243	51.191
128	102.366	102.366
256	204.715	204.834
512	409.600	409.463
1024	819.268	818.654
2048	1638.127	1637.444
4096	3275.435	3275.162

The experiment results show that both ESP and Q-ESP have almost the same throughput for the same packet size. Someone may expect that ESP throughput Exceed Q-ESP throughput; as Q-ESP has larger header and as its authentication cover larger area. However, it is not the case because, we found that in most cases, pads are added to the end of the messages to complete block size and in the average processing we found that ESP and Q-ESP nearly have the same throughput. From Table II results, we can deduce that in Best-Effort environment (I.e., without QoS), we can use Q-ESP instead of ESP without affecting the performances.

B.Priority Control Experiment

In order to provide accurate measurement for Q-ESP performance, and to demonstrate Q-ESP capabilities in supporting QoS active admission control; we setup a testbed to measure the throughput of Q-ESP with QoS priority treatment, against ESP and Q-ESP without QoS priority enforcement.

Fig. 8 shows the used network configuration. The testbed configuration consists of two domains connected to each other through simple differentiated services cloud. Each domain contains three load PCs (PC 2 to 4) for generating traffic and creating congestion situations, and one protected PC (PC1). The Q-ESP/ESP gateway protects only PC1 by providing encryption and authentication services to its traffic.

MGEN and Iperf were installed in all PCs. The differentiated services cloud contains two edge routers to perform active admission control activities (such as classifying packets and setting priorities), and QoS core routers to handle traffic differently according to its priority.

To perform our tests, MGEN has been used in PC1 to generate small UDP packet sizes in order to simulate a real-time flow (VoIP). More precisely, to simulate simultaneous communications we have generated streams with different frequencies (100... 1000 packets per second). These packets

were cryptographically protected by Q-ESP / ESP gateways.

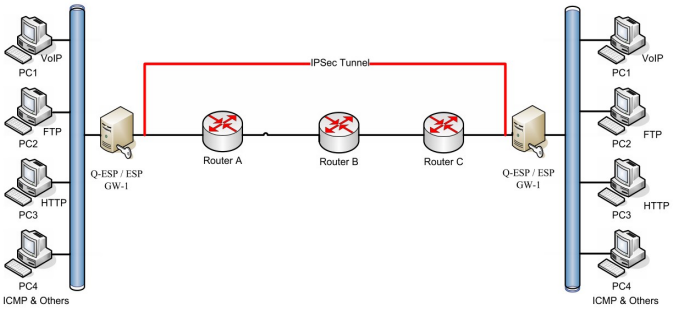

Fig. 8. Testbed for priority control experiments.

The PCs for the load traffics have been used to generate different types of flows (FTP, HTTP, UDP and ICMP) with different packets sizes. These flows are intended to simulate real network traffic.

We implemented two scenarios. In the first scenario, we measured the performance of ESP and Q-ESP in case of network congestion without QoS priority treatment.

In the second scenario, the same previous experiments have been conducted while setting priorities and enforcing admission control. More precisely, the edge router is configured to treat Q-ESP traffic as high priority traffic and the other three load traffic as low-priority traffic. Congestion is created and the measurement of Q-ESP traffic stream is recorded. We performed these tests using AES for the encryption and HMAC-SHA1 for the authentication.

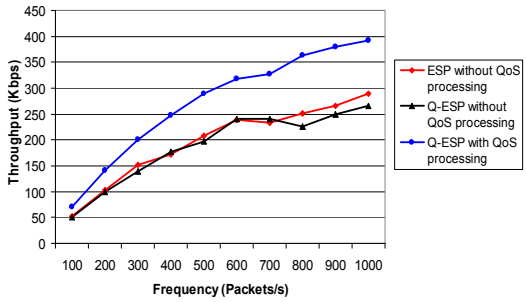

Fig. 9. Priority control experiment results

Fig. 9 shows that Q-ESP with QoS priority treatment has higher throughput in comparison with ESP and Q-ESP without QoS treatment. Actually, the throughput increases when packet size and/or frequency increase. For 128 byte packet size sent with frequency 1000 packet per second, the throughputs for ESP and Q-ESP without QoS priority treatment, and Q-ESP with QoS priority enforcement are 289.587, 265.198 and 391.560 kbps respectively. Clearly, in congestion situations the throughput of ESP and Q-ESP in the first scenario was degraded due to the impact of other competing traffics which try to consume the shared bandwidth. In the second scenario, the Q-ESP was assigned a specific queue with a reserved bandwidth. Thus, Q-ESP throughput can increase as long as it does not exceed the allocated bandwidth.

From the previous tests, we can deduce that Q-ESP almost gives the same performance like ESP in best effort environment, while it supports active admission control in QoS managed environment.

To conclude, Q-ESP offers the same security services as IPSec AH and ESP used conjointly, while it also allows QoS processing.

VII. Conclusion

In this paper, we presented a new security protocol "called Q-ESP" which: (1) supports QoS in doing active admission control, and (2) combines the security features of both the IPSec ESP and AH protocols. We gave details of the structure and the processing steps of our Q-ESP protocol. Moreover, we presented some information about our NetBSD kernel-based implementation. Finally, we presented our performance evaluation experiments. Based on the obtained results, we can conclude that: in Best-Effort environment, ESP and Q-ESP throughput are mainly equivalent, while in QoS environment; Q-ESP is notably better as it enables performing QoS easily while guaranteeing secure communications. To do that, Q-ESP carries out tunneling while enabling control equipment examining the upper layer headers (e.g., TCP / UDP), and thus provide QoS.

Now, we are proposing an IETF draft and implementing an IPv6 support for Q-ESP. Besides that, to enable dynamic configuration of Q-ESP VPN tunnels, we are updating the IKE (Internet Key Exchange) daemon (with Racoon).

References

[1] Kent, S., Atkinson, R., Security architecture for the Internet protocol. IETF, RFC2401, Nov. 1998.

[2] Borg, N., Savanberg, E., Schelén, O., "Efficient Multi-Field Packet Classification for QoS Purposes"; 1999; International Workshop on Quality of Service '99; p. 109-118.

[3] Gupta, P. "Algorithms for routing lookups and packet classification,". PhD. Thesis, Stanford University, Stanford, CA, December 2000.

[4] Huston, G., "Next Steps for the IP QoS Architecture", RFC 2990, November 2000.

[5] Kent, S. IP Encapsulating Security Payload (ESP). IETF, RFC4303, December. 2005.

[6] Kent, S. IP authentication header. IETF, RFC4302, December. 2005.

[7] Madson, C., Glenn, R., The use of HMAC-SHA-1 within ESP and AH. IETF, RFC2404, Nov. 1998.

[8] Frankel, S., Glenn, R., Kelly, S., "The AES-CBC Cipher Algorithm and Its Use with IPSec", RFC3602, September 2003.

[9] NetBSD 5.0 Release Candidate 2 -Feb,10 2009. available at: <ftp://ftp.netbsd.org/pub/NetBSDdaily/netbsd-5-0-RC2/ >.

[10] Samuel J. Leffler, Errno Consulting. Fast IPSec: A High-Performance IPsec Implementation. USENIX Association, BSDCon '03. San Mateo, CA, USA. September 8–12, 2003.

[11] Wireshark available at <http://www.wireshark.org>.

[12] Nikov, V. (2006) 'A DoS Attack Against the Integrity-Less ESP (IPSEC)', International Conference on Security and Cryptograph, p. 192-199.

[13] Elkeelany, O. Matalgah, M.M. Sheikh, K.P. Thaker, M. Chaudhry, G. Medhi, D., Qaddour, J., "Performance analysis of IPSec protocol: encryption and authentication", IEEE Communications Conference (ICC 2002), 2002, p. 1164–1168.

[14] The Multi-Generator (Mgen) available at: <http://cs.itd.nrl.navy.mil/work/mgen/>

[15] D.Mills, "Network Time Protocol version 2 specification and implementation", RFC 1119, September 1989.

[16] Iperf tool, available at : <http://dast.nlanr.net/Projects/Iperf/>.