ANALYTICAL MODEL OF IPSEC PROCESS THROUGHPUT

Adam TISOVSKY[1], *Ivan BARONAK*[1]

[1]Institute of Telecommunications, Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology, Ilkovicova 3, 812 19 Bratislava, Slovak Republic

tisovsky@ut.fei.stuba.sk, baronak@ut.fei.stuba.sk

Abstract. *The paper concerns with throughput of securing process, which cannot be described neither by a constant value of bits per second nor by a constant value of packets per second over the range of packet sizes. We propose general throughput model of IPsec process based on characteristic parameters that are independent on the packet size. These parameters might be used for a comprehensive definition of throughput on any security system. Further, a method for obtaining characteristic parameters is proposed. Usage of the method can significantly decrease count of throughput measurements required for modelling the system.*

Keywords

Architecture, IPsec, model, security, throughput.

1. Introduction

IPsec standard is one of widely deployed mechanisms for securing the network traffic – it is a suite of protocols, standards and rules ensuring data integrity, authenticity and confidentiality [1], [2]. Process of securing network communication is computationally intensive and being handled by a network device can bring degradation to qualitative parameters of the network, mainly increase of delay and decrease of throughput [3], [4]. It is also a big challenge to fulfil continuously heightening level of security when the amount of secured traffic is increasing concurrently. For this reason, it is important to examine the performance of security systems for various security configurations and types of traffic.

Specific property of securing process is that its throughput is dependent on size of the packet. The throughput cannot be expressed neither by a constant value of bits per second, nor by a constant value of packets per second over the whole range of packet sizes. On the contrary, over this range it will have non-constant and nonlinear trend, as is depicted in Fig. 1. This is in contrast with performance of other common processes in packet networks, like bit rate of link interfaces or packet rate of switching and routing.

This introduces difficulties into modelling performance parameters of security systems when the value of service rate needs to be known, e.g. dimensioning throughput of mixed traffic comprising of more packet sizes, modelling a queuing delay or packet loss. In these cases, we need to know values of throughput for every packet size that is present in the traffic. In doing so, performing a lot of measurements could be rather time-consuming and inconvenient.

Aim of this paper is to propose a model of throughput of securing based on parameters independent on the packet size. The model should be valid for a wide range of IPsec system implementations and should provide reliable results without the need for detailed knowledge of the system internal architecture.

Model synthesis is based on the analysis of software and hardware components of the system – determined are operations that create potential bottleneck process in the system and consequently these operations are joined into groups according to their dependency on the packet size. As a result, throughput of securing process will be expressed by *characteristic parameters* that are independent on the packet size. This formulation further allows calculation of throughput for any packet size – a method for obtaining characteristic parameters will be proposed. Lastly, the model verification will be performed for various implementations of IPsec systems.

Although the paper focuses on IPsec, presented principles and methods are general and therefore should be applicable for any other security protocol.

2. Related Works and Motivation

Significant amount of works focused on examination of security system performance is dealing with experimental evaluation of the system performance. Authors rarely research possibilities of modelling or mathematical formulation of performance of security system. Authors in [5] present mathematical model of throughput of symmetric algorithms, where input variables are a number of instructions needed to encrypt one block of data and number of instructions executed in one processor clock cycle. In [6] authors study size overhead and processing overhead of IPsec, however, they consider only algorithmic time requirements, and not the time requirements of protocol processing that are constant for the packet of any size.

To our knowledge no literature provides analytical model of time requirements or throughput of entire IPsec process (or other security protocol) addressing its dependency on packet size, which is based on analysis of hardware and software components of the system.

In the field of security systems, many authors consider enhancing performance of the security system as one of the key tasks. The goal can be achieved either in a layer of hardware – especially working with configurable processors of type FPGA (Field-Programmable Gate Array), which can be programmed and optimized for execution of cryptographic algorithms, security protocol or both. Processors with configured logic provide higher performance than processors of general purpose [3], [7]. Increase of performance can be achieved also by lowering communication overhead and better cooperation between components in a layer of software, which are represented mainly by device drivers, protocol stacks and cryptographic framework. Different implementations of one security protocol on the same device usually provide different performance [4].

These are the main reasons of variability of different IPsec system solutions. A model proposed in this paper must therefore address this variability in order to be considered as general.

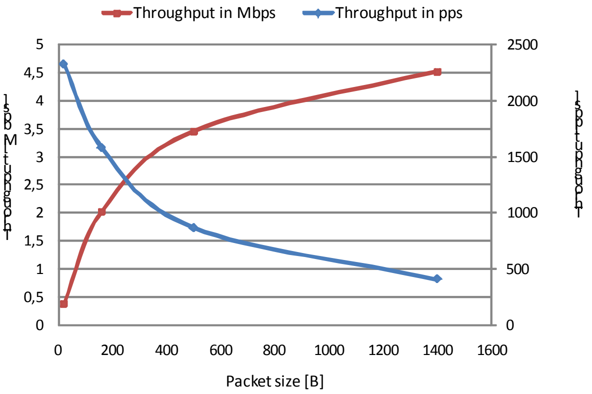

Fig. 1: Throughput of IPsec process over the range of packet sizes performed by Cisco 1841 ISR.

3. IPsec System

In this section, analysis of hardware and software components of IPsec system is performed. Essential questions are – which operations form the IPsec process, which are potential bottleneck in the system, i.e. which are executed in serial and which in parallel, and how is each operation dependent on size of the packet.

3.1. Hardware Architecture

In Fig. 2 is shown general hardware architecture of IPsec system [8], [9], [10], [11]. Most important components are: main processor (CPU), security processor, which is optional, main memory (also called system or kernel memory), L2 caches, network adapters, buses and direct communication channels.

In modern systems each processing unit access to the main memory through a direct communication channel called DMA (Direct Memory Access) when no control and management is required from the main processor. For instance, a network adapter moves the packet after its receiving on the interface “silently” to the main memory, in parallel to operations executed by the main processor, and then sends information to the main processor (*interrupt request*). Similarly, security processor reads and writes data to the main memory through its direct memory access without assistance of the main processor.

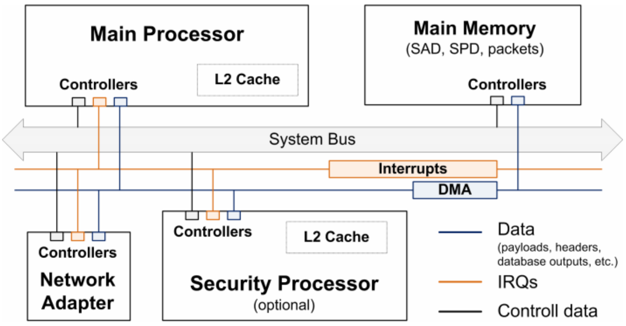

Fig. 2: Hardware architecture of IPsec system.

Interrupt requests (IRQ) are asking the processor to interrupt current operation and execute operation of a higher priority. They are a common method to provide communication between independent processors operating in parallel, asynchronously. Information about finishing an operation or a request for a new operation is sent immediately without waiting. This approach lowers packet sojourn time in the system, but on the contrary it increases processing overhead by storing and restoring data from the interrupted process (*context-switching*).

Because the process of securing is computationally intensive, a separate security processor can be added to the system to increase performance of the system and to offload the main processor. Security processor is optimized for executing special operations, commonly for acceleration of cryptographic algorithms (in this case it is called *cryptographic accelerator,* which operates in *look-aside* mode), but also the whole protocol processing can be accelerated (then it is called *security unit*, which is more complex than accelerator and operates in *flow-through* mode). Usually, FPGA processors are employed as security processors because of their configurability and ease of optimization.

Placement of security processor within the device can be done either as a separate card connected via PCI bus, or as a processor embedded on the board, or as a part of SoC (System-on-a-chip), which composes of more processing cores.

3.2. Software Architecture

Software equipment is represented by an operating system, which includes implementation of the IPsec standard as well. Software determines how effectively will be resources provided by the hardware utilized. In Fig. 3 is shown general software architecture of a security system. This architecture follows the main concepts native for operating systems Linux 12 and BSD [13], [14] and which consider also leading producers of security hardware and software (Intel [11], Cavium [15], Freescale [8]).

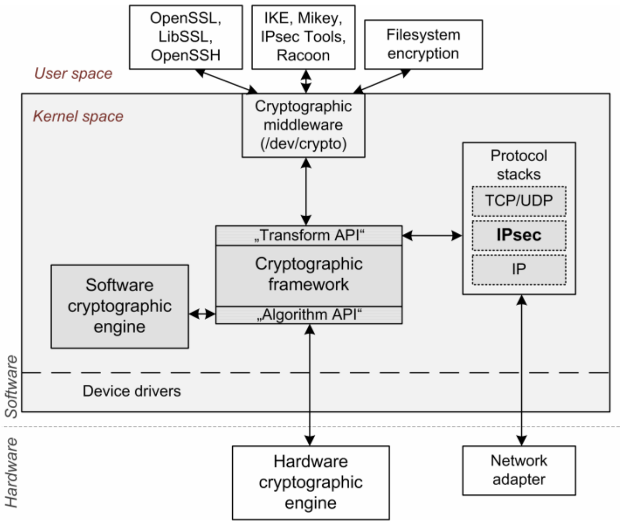

Fig. 3: Software architecture of IPsec system.

When speaking particularly of IPsec system, significant part of operations is executed in *kernel-space*. Only protocol IKE (Internet Key Exchange) is processed in *user-space*, however, it is executed only in a set-up phase of the connection and therefore it has no influence on the throughput. On the contrary, execution of other security protocols, e.g. SSL/TLS (Secure Socket Layer/Transport Layer Security) and SRTP (Secure Real-time Transport Protocol) is performed in user-space.

1) IPsec Stack

In kernel-space are executed all protocol operations defined by IPsec standard for securing the traffic. The protocols are AH (Authentication Header) and ESP (Encapsulating Security Payload), i.e. the operations include a creation of protocol headers and entries and look-ups in databases SPD (Security Policy Database) and SAD (Security Association Database). Database SPD is used for determining whether the packet is IPsec or not, and if so, security configuration for the packet is looked-up in the SAD database. Both databases are part of the main memory.

Most important fact is that mentioned AH and ESP protocol operations do not work with packet payload at all. This means that time requirements of these operations are independent on the packet size.

2) Cryptographic Framework

Cryptographic framework makes cryptographic operations performed by cryptographic engine (either software or hardware) available to all components of the kernel-space, including IPsec stack. It defines two interfaces – first for access of the components to framework, second for access of the framework to cryptographic algorithms (*Transform API* and *Algorithm API* in Linux systems, *Consumer API* and *Producers API* in BSD systems). It further builds up the transformation configuration (fetches requests and creates their descriptors, creates pointers for payload, keys and configuration data) and calls driver of cryptographic engine.

In case the system is not equipped by separate hardware cryptographic engine (cryptographic accelerator), framework calls the algorithm from its own library using pseudo-driver *cryptosoft*. Comparing to driver of hardware engine pseudo-driver is very trivial and introduces low or even negligible processing overhead. In case that hardware cryptographic engine is present, the cryptographic operations are moved to the layer of hardware.

Based on description of operations of cryptographic framework we can conclude that they are independent on the size of the packet – they are not working with packet payload as was also the case of protocol operations.

3) Cryptographic Algorithms

Cryptographic algorithms transform plain-text payload to a cipher-text. They can either encrypt the payload to ensure data confidentiality, e.g. symmetric encryption algorithms 3DES or AES, or they can compute authenticated hash to ensure data integrity and authenticity, e.g. authentication algorithms MD5-HMAC or SHA-1-HMAC.

Most of the algorithms process the data in blocks of fixed size – padding is added if payload size is not a multiple integer of the block size. Encryption algorithms may work in various modes – either without feedback (ECB – Electronic Codebook), or with feedback between consecutive blocks (e.g. CBC – Cipher-Block Chaining, CFB – Cipher Feedback Mode). Most important note is that in every mode the data are processed without a nonlinear feedback, i.e. feedback modes reflect only preceding block of data. This means that time requirements of raw cryptographic operations are dependent on packet size and are directly proportional to number of blocks in the packet.

Another note is that entire algorithmic processing composes besides mentioned cryptographic operations also from a small overhead introduced by algorithm initialization – mainly derivation of *round keys* from the main key. Time requirement of this overhead is for every packet the same – it does not depend on the number of blocks in the packet.

4) Modes of Operation

The main processor and cryptographic engine cooperate in one of two modes. Mode of operation has a decisive impact for defining operations which create the bottleneck process. The mode indicates whether cryptographic engine operates towards the main processor in serial (*synchronous mode*) or in parallel (*asynchronous mode*).

Synchronous mode of operation – in this mode the main processor has to wait until algorithm is finished. In case that cryptographic engine is only software implementation, i.e. algorithm is executed by the main processor, synchronous mode is the only possible mode. If cryptographic engine is a separate hardware processor, this mode may, however, lower potential performance of the system. The main processor waits idly until algorithm is finished and is blocked for any other process. On the other hand, synchronous mode of operation introduces lower communication overhead between processing units, i.e. driver of cryptographic engine is simpler and interrupt requests are not raised. Synchronous mode of operation is more likely to be found in the security system implementations [4] – it is employed if:

- the device is not equipped by hardware cryptographic engine (cryptographic operations are executed by the main processor),
- hardware cryptographic engine is incapable of the asynchronous operation (lower-end devices),
- cryptographic framework is not capable of the asynchronous operation (native cryptographic framework in Linux, called Cryptographic API),
- hardware cryptographic engine is not supported by the cryptographic framework (driver is missing),
- synchronous mode is chosen administratively (if communication overhead is greater than benefits introduced by asynchronous mode).

Asynchronous mode of operation – this mode is suitable only for a system equipped by a hardware cryptographic engine. The main processor requests engine to perform cryptographic operations. Employing *call-back function* it does not wait for their finishing, however, it returns for executing its own protocol operations. Cryptographic engine and the main processor are able to operate in parallel, then. The bottleneck process will be one of two separate processes. As this form of communication is more complex, communication overhead is also higher than in synchronous mode (interrupt requests are present). Asynchronous mode of operation is employed only if both cryptographic framework and cryptographic engine support it and if choice of this mode is advantageous.

3.3. Summary

Based on the preceding analysis of hardware and software components of IPsec system, we can deduce that the only determinant factor in formulating throughput of IPsec system will be the mode of operation. All other possibilities for enhancing the system performance will be in lowering communication overhead, i.e. lowering amount of transferred control and configuration data, in efficient entries into memories and in acceleration of cryptographic algorithms. The enhancements would lead into lower time requirements of the particular operation, but would not eliminate the operation entirely.

4. Model of IPsec Process Throughput

In order to delimit the operations that cause the bottleneck process we take a look at a chain of operations that packet undergoes when it passes through the system. Presented operation chain follows principles of operation systems Linux and BSD, and also principles presented by several vendors, e.g. Freescale [16], Intel [17], Elliptic [18], Cisco [19] and Mikrotik [20].

4.1. Synchronous Mode of Operation

In Fig. 4 is shown chain of operations for synchronous mode of operation. Description of these operations is following:

1. Processing begins after receiving a frame on the network interface. Network adapter, which operates towards the main processor asynchronously as a separate processing unit, checks the frame for errors, executes operations of the data link layer, removes the frame header and transfers a packet to the main memory (*RX Ring Buffer*) via DMA. All these operations are performed independently on the main processor (i.e. in parallel), so they do not participate on the potential bottleneck process.
2. When the transfer of the packet to memory is finished, network adapter sends to the main processor interrupt request (IRQ), which must be served immediately. Processor launches *interrupt handler* or *interrupt service routine* (ISR), which is actually a network adapter driver that fetches information about received packet. It schedules also SoftIRQ program in which protocol operations are executed.
3. Packet header may be stored in L2 cache of the main processor what speeds up its processing. It is checked for errors and against the SPD to determine whether the packet falls into IPsec policy.

4. If packet falls into IPsec policy, security configuration in SAD is looked up.
5. If ESP protocol is used, *ESP trailer* containing padding bits is added to the packet.
6. Cryptographic framework is launched. It is responsible for building up transformation configuration, i.e. fetching requests, creating their descriptors and creating pointers for payload, keys and configuration data.
7. This information is sent to cryptographic engine using the driver. If cryptographic engine is a software implementation, the driver is very trivial.
8. Cryptographic engine reads transformation configuration from the main memory. If round cryptographic algorithm is used, round keys are derived from the main key.
9. Cryptographic engine reads plain-text data assigned for securing from the main memory.
10. Cryptographic operations are performed.
11. Cipher-text is written to the main memory.
12. The main processor finishes protocol processing of the packet, i.e. adds AH and/or ESP header.
13. In case of IPsec tunnel mode, a new IP header is created.
14. The rest of operations of the network layer is performed, e.g. routing, QoS policy, etc.
15. Information about placement of the packet in memory and request for its transmission is sent do the network adapter.
16. Network adapter creates frame header and transmits the frame.

When speaking of process as a set of operations executed by one or more processors in a synchronous manner it is evident that process of securing consists of all operations described above except from the ones executed by the network adapter. We can divide these operations of securing process according to their dependency on the packet size into two groups:

1. Time requirements of operations of the first type are independent on the packet size. In Fig. 4, these are operations with orange background. They are aforementioned operations of protocol stacks, i.e. manipulation with headers and databases look-ups, building up transformation configurations, executing drivers and deriving round keys. These operations do not work with packet payload at all. Computational rate of these operations could be expressed in packets per second independently on the packet size.

   Now, we join these operations into one sub-process. Its time requirements will be expressed as

   $$t_{fix} = t_{IP} + t_{IP\sec} + t_{com} + t_{key}\,, \qquad (1)$$

   where t_{IP} are the time requirements of IP protocol stack, $t_{IP\sec}$ are the time requirements of IPsec protocol stack and cryptographic framework, t_{com} is communication overhead (mainly execution of drivers) and t_{key} is the algorithm overhead (mainly derivation of round keys).

2. On the contrary, time requirements of the second type operations are dependent on the packet size – they are directly proportional to the number of algorithmic data blocks in the packet. In Fig. 4, they are the ones with purple background. Except from the raw cryptographic operations also read/write operations of plain-text/cipher-text from and to the main memory belong here. Computational rate of these operations can be expressed by a constant bit rate independent on the packet size. We join these operations in one sub-process. Its time requirements will be expressed as

   $$\frac{L_{alg}}{R_{alg}} = \frac{L_{alg}}{R^{*}_{alg}} + \frac{L_{alg}}{R_{rw}}\,, \qquad (2)$$

   where L_{alg} is the size of secured data, R_{alg} is bit rate of the sub-process, R^{*}_{alg} is bit rate of raw cryptographic operations and R_{rw} is bit rate of read/write operations of plain-text/cipher-text.

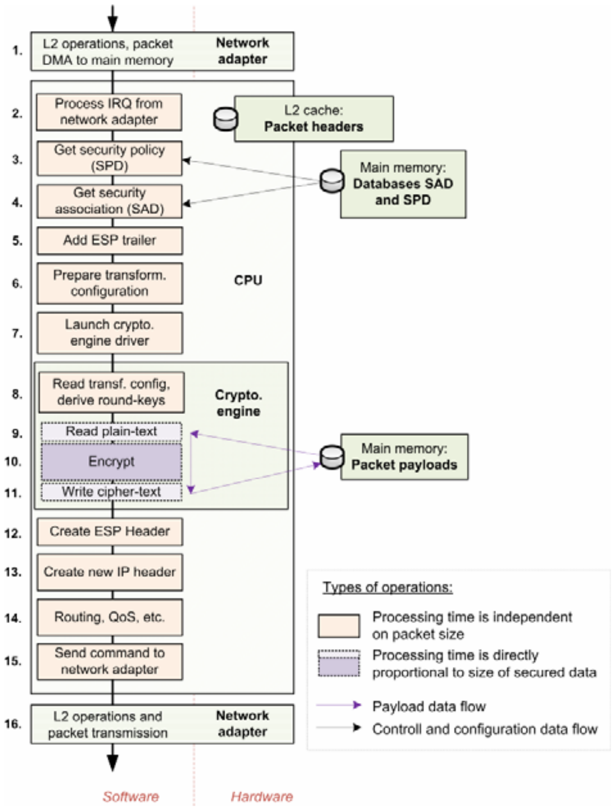
1.
L2 operations, packet DMA to main memory
Network adapter
2.
Process IRQ from network adapter
L2 cache: Packet headers
3.
Get security policy (SPD)
Main memory: Databases SAD and SPD
4.
Get security association (SAD)
5.
Add ESP trailer
6.
Prepare transform. configuration
CPU
7.
Launch crypto. engine driver
8.
Read transf. config, derive round-keys
Crypto. engine
9.
Read plain-text
10.
Encrypt
Main memory: Packet payloads
11.
Write cipher-text
12.
Create ESP Header
13.
Create new IP header
14.
Routing, QoS, etc.
15.
Send command to network adapter
16.
L2 operations and packet transmission
Network adapter
Types of operations:
Processing time is independent on packet size
Processing time is directly proportional to size of secured data
Payload data flow
Controll and configuration data flow
Software
Hardware

Fig. 4: Chain of operations in IPsec processing for synchronous mode of operation.

Since both sub-processes are synchronous, time requirements of the whole process of securing will be the sum of time requirements of individual sub-processes, i.e.

$$t_S = t_{fix} + \frac{L_{alg}}{R_{alg}} . \tag{3}$$

Throughput of securing process will be inversed value of its time requirements, i.e. $R_s = 1/t_S$ packets per second.

From aforementioned relation, it is evident why throughput of securing cannot be expressed neither by a constant bit rate, nor a constant packet rate over the whole range of packet sizes. However, parameters t_{fix} and R_{alg} describe the process of securing comprehensively over the whole range of packet sizes independently from the packet size, and therefore we name them *characteristic parameters* of securing process.

4.2. Asynchronous Mode of Operation

In Fig. 5 is shown chain of operations for asynchronous mode of operation. Description of these operations is the same as in the previous case. The difference is in parallelism of operation of the main processor and the cryptographic engine. As a consequence, bottleneck process may arise from one of two processes – either protocol, framework and driver operations executed by the main processor, or communication and algorithmic operations executed by the cryptographic engine, depending on which one has the higher time requirements.

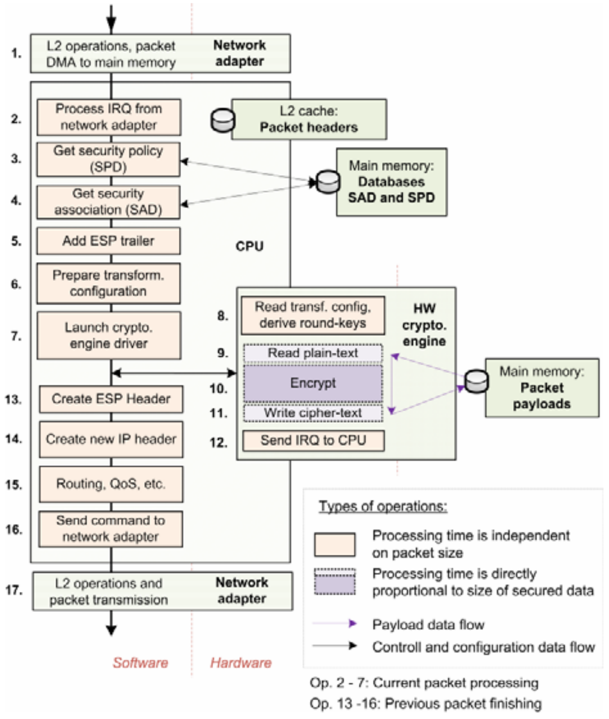
1.
L2 operations, packet DMA to main memory
Network adapter
2.
Process IRQ from network adapter
L2 cache: Packet headers
3.
Get security policy (SPD)
Main memory: Databases SAD and SPD
4.
Get security association (SAD)
5.
Add ESP trailer
CPU
6.
Prepare transform. configuration
7.
Launch crypto. engine driver
8.
Read transf. config. derive round-keys
HW crypto. engine
9.
Read plain-text
Encrypt
10.
Main memory: Packet payloads
11.
Write cipher-text
12.
Send IRQ to CPU
13.
Create ESP Header
14.
Create new IP header
15.
Routing, QoS, etc.
16.
Send command to network adapter
17.
L2 operations and packet transmission
Network adapter
Software
Hardware
Types of operations:
Processing time is independent on packet size
Processing time is directly proportional to size of secured data
Payload data flow
Controll and configuration data flow
Op. 2 - 7: Current packet processing
Op. 13 -16: Previous packet finishing

Fig. 5: Chain of operations in IPsec processing for asynchronous mode of operation.

Time requirements of securing process in the system with asynchronous mode of operation will be then

$$t_S = \max\left(t_{fix},\ t_{oh} + \frac{L_{alg}}{R_{alg}} \right), \tag{4}$$

where t_{fix} are the time requirements of operations executed by the main processor, t_{oh} is communication and key derivation overhead performed by cryptographic engine (operations 8 and 12 in Fig. 5), and L_{alg} / R_{alg} are the time requirements of algorithmic operations including read/write operations. Securing process in asynchronous mode of operation will be described by three characteristic parameters: t_{fix}, t_{oh} and R_{alg}.

4.3. Size of Secured Data

IPsec in tunnel mode secures entire original IP packet [21], [22]. In rough calculations L_{alg} can be considered as size of the original IP packet. In precise calculations L_{alg} will be calculated using following formulas. Size of secured data for AH authentication will be

$$L_{alg}^{AH} = \left\lceil \frac{L + L_{IP} + L_{AH}}{B_{alg}} \right\rceil B_{alg} , \tag{5}$$

where L is the size of the original IP packet, L_{AH} is the size of AH header, L_{IP} is the size of authenticated fields in new IP header (*non-mutable fields*) and B_{alg} is a block size of the algorithm. In case of IPv4, L_{IP} equals 12 bytes, in case of algorithms MD5-96 a SHA-1-96 L_{AH} equals 24 bytes (12 bytes for fixed fields and 12 bytes for a hash) and B_{alg} equals 64 bytes.

Size of secured data for ESP encryption will be following:

$$L_{alg}^{ESP} = \left\lceil \frac{L + 2}{B_{alg}} \right\rceil B_{alg} , \tag{6}$$

where L is the size of the original IP packet and B_{alg} is the block size of the algorithm. Addition of “+2” in the numerator is because ESP trailer contains mandatory fields *Next Header* and *Pad Length* of size 2 bytes. B_{alg} equals 8 and 16 bytes in case of algorithms 3DES and AES, respectively.

4.4. Summary

Based on performed analysis, we can suppose that various implementations of IPsec system introduce only variability in relative significance of proposed characteristic parameters, and do not mean elimination or addition of a new parameter. The only determinant factor in formulating throughput will be the mode of operation. Proposed model shall be then valid for any implementation of IPsec system. The validation for various implementations is presented in the Section 7 of this paper.

5. A Method for Obtaining Characteristic Parameters

In this section, a method for obtaining characteristic parameters will be presented. The method presumes that securing process is the bottleneck process in the system. In such case throughput of securing process (R_S) will be also an end-to-end throughput (R_M). And vice versa, if we measure end-to-end throughput, we obtain throughput of the securing process. Thus, we can state: $R_S = R_M$. This will transform to the time domain as

$$\frac{L_M}{R_S} = \frac{L_M}{R_M} = t_S = t_M\,, \tag{7}$$

what is the time required to process one packet of size L_M by the bottleneck process (t_S) as well as the time interval between arrivals of two packets at the recipient's site (t_M). We assume that no packet loss is caused by a random overflow of the packet buffer.

Aforementioned relation means that characteristics of securing process given by parameters t_{fix} and R_{alg} (3), eventually also by t_{oh} (4), will be "mirrored" into the measured throughput. Measured throughput will therefore provide enough information to evaluate the characteristic parameters.

5.1. Synchronous Mode of Operation

If we have two unknowns (t_{fix} and R_{alg}) we can use system of two equations to evaluate them. Known variables in the first equation will be the throughput R_{M1} measured for packet size L_{M1}, size of secured data L_{alg1} in this packet, and in analogue, values for another packet size in the second equation. The system of two equations in two unknowns will be then

$$\frac{L_{M1}}{R_{M1}} = t_{fix} + \frac{L_{alg1}}{R_{alg}}\,, \tag{8}$$

$$\frac{L_{M2}}{R_{M2}} = t_{fix} + \frac{L_{alg2}}{R_{alg}}\,. \tag{9}$$

Solving the system we get expressions of the characteristic parameters:

$$R_{alg} = \frac{(L_{alg2} - L_{alg1})R_{M1}R_{M2}}{L_{M2}R_{M1} - L_{M1}R_{M2}}\,. \tag{10}$$

$$t_{fix} = \frac{L_{M1}}{R_{M1}} - \frac{L_{alg1}}{R_{alg}}\,. \tag{11}$$

Thereafter we are able to calculate estimated throughput R_{calc} for any size of packet L_{calc} by a relation

$$R_{calc} = \frac{L_{calc}}{t_{fix} + \frac{L_{alg}}{R_{alg}}}\,, \tag{12}$$

where L_{alg} is size of secured data in this packet.

As can be seen, besides the possibility of estimating throughput we get also information about two sub-process in the securing process that are described by values t_{fix} and R_{alg}.

5.2. Asynchronous Mode of Operation

Similar assumptions and principles as in the previous case are standing also for asynchronous mode. The difference is that securing process is divided into two separate independent processes whereas only one of them becomes a bottleneck process for a particular packet size.

If time requirements of operations executed by the main processor are higher than requirements of operations executed by the cryptographic engine, then throughput of the system will be given by a relation

$$R_{calcA} = \frac{L_{calc}}{t_{fix}}\,, \tag{13}$$

where L_{calc} is the packet size, which is the throughput calculated for and t_{fix} are the time requirements independent on the packet size.

Contrary, if time requirements of operations executed by the cryptographic engine are higher, then throughput of the system will be given by a relation

$$R_{calcB} = \frac{L_{calc}}{t_{oh} + \frac{L_{alg}}{R_{alg}}}\,, \tag{14}$$

which is derived from the system of two equations in two unknowns in the same manner as it being in case of synchronous operations, but instead parameter t_{fix} here appears parameter t_{oh} (see Fig. 5).

We will suppose that throughput of securing process will be given by (13) for the small packets and by

(14) for the large packets. In such case, three measurements of throughput are necessary. First measurement will be performed for a small packet to obtain parameter t_{fix}, two other measurements will be performed for the large packets to obtain parameters t_{oh} and R_{alg}. Correctness of the last two measurements will be verified so that these values have to be different from values calculated by (13).

Throughput breakpoint, i.e. packet size L_{bp} when both processes require the same amount of time, can be derived from equality of (13) and (14), i.e.:

$$L_{bp} = R_{alg}\left(t_{fix} - t_{oh}\right). \qquad (15)$$

Compacted expression of throughput of securing process over the whole range of packet sizes for asynchronous mode of operation will be then

$$R_{calc} = \begin{cases} R_{calcA} & \text{for } L_{min} < L_{alg} \le L_{bp} \\ R_{calcB} & \text{for } L_{bp} < L_{alg} \le L_{max} \end{cases}, \qquad (16)$$

where L_{min} is the smallest possible size of secured data and L_{max} is the highest possible size of secured data in the packet.

5.3. Usage of the Method

Usage of the method eliminates the need for performing a lot of measurements in order to create throughput profile for the whole range of packet sizes. For instance, document RFC 2544 recommends performing measurements for packet sizes of 64, 128, 256, 512, 1024, 1280 and 1420 bytes. Moreover, in modelling performance of security system we need to know throughput values for each packet size that is present in the traffic. The measurements can be rather time-consuming as one measurement can last several minutes using UDP iterative search technique recommended in RFC 2889. Proposed method can be used also in a situation when we cannot perform any measurement, but we know throughput values for two different packet sizes, for instance from a technical specification of the device. For these reasons, characteristic parameters could be practically used for comprehensive and convenient definition of securing process throughput on any security system.

6. Experimental Verification

Experimental verification was performed on a test-bed illustrated in Fig. 6. Two routers Cisco 1841 ISR with synchronous mode of operation acted as IPsec gateways. Other parameters of the test were following:

- IPsec in tunnel mode,
- Iperf measuring tool,
- UDP transport protocol, iterative search algorithm recommended in RFC 2889,
- 20 second duration of each iteration,
- 5 repetitions of each test to calculate the average.

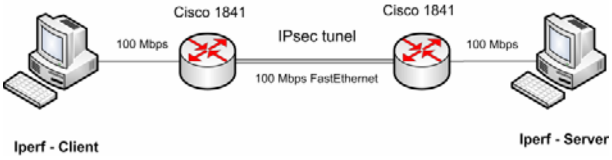

Fig. 6: Experimental environment setup.

In Tab. 1 are listed calculated values of t_{fix} and R_{alg} for various security combinations using (10) and (11) with throughput measured for packet sizes 128 bytes and 1280 bytes. Bit rate R_{alg} represents theoretical limit of the throughput for very large packets and inversed value of time t_{fix} represents the limit of the throughput for very small packets. It can be noted that value t_{fix} varies a little between encryption or authentication algorithms what is caused by the different algorithm overheads – round key derivations. For instance, in case of algorithm AES the value increases by about 0,02 ms with every increase of the key by 64 bits.

In Tab. 2 are confronted measured and calculated values of throughput using (12) for packet sizes 64, 512 and 1420 bytes. The difference is expressed as $\Delta = \left|\frac{R_M - R_{calc}}{R_M}\right| \cdot 100$ [%]. In all cases, it keeps below 2 %.

Tab.1: Calculated characteristic values t_{fix} and R_{alg}.

PROTOCOL -algorithm	Measured throughput (R_{M1} = 128 B) [Mbps]	Measured throughput (R_{M2} = 1280 B) [Mbps]	t_{fix} [ms]	R_{alg} [Mbps]
AH-sha1	2,003	12,919	0,4761	32,749
AH-md5	2,152	14,838	0,4489	42,986
ESP-des	2,233	8,965	0,3778	13,483
ESP-3des	1,712	4,399	0,3939	5,323
ESP-aes128	2,204	11,382	0,4098	21,179
ESP-aes192	2,079	10,285	0,4296	18,316
ESP-aes265	1,987	9,576	0,4472	16,630

Tab.2: Comparison of measured and calculated throughput for different packet sizes and security configurations.

PROTOCOL-algorithm	Packet size [bytes]	Measured throughput R_M [Mbps]	Calculated throughput R_{calc} [Mbps]	Difference Δ
AH-sha1	64 512 1420	1,021 6,870 13,691	1,033 6,769 13,788	0,9 % 1,6 % 0,6 %

AH-md5	64 512 1420	1,086 7,618 15,800	1,103 7,484 15,912	1,5 % 1,8 % 0,7 %
ESP-des	64 512 1420	1,235 6,021 9,233	1,217 5,967 9,289	1,4 % 1,1 % 0,6 %
ESP-3des	64 512 1420	1,002 3,534 4,485	1,019 3,489 4,508	1,2 % 1,3 % 0,5 %
ESP-aes128	64 512 1420	1,613 6,771 11,959	1,625 6,718 11,983	0,7 % 0,8 % 0,2 %
ESP-aes192	64 512 1420	1,097 6,222 10,738	1,102 6,204 10,803	0,5 % 1,1 % 0,6 %
ESP-aes256	64 512 1420	1,053 5,921 9,904	1,056 5,851 10,004	0,3 % 1,2 % 1,0 %

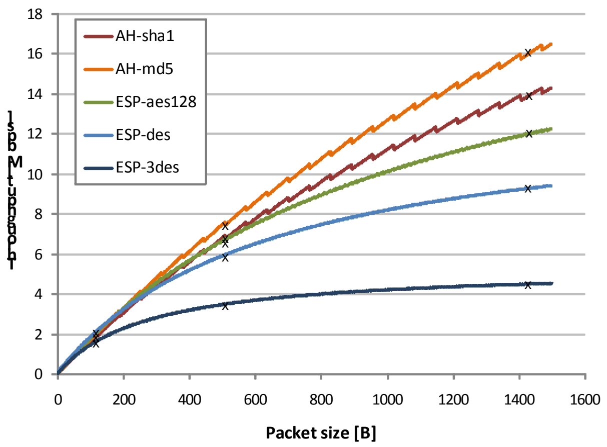

Fig. 7: Calculated throughput over the whole range of packet sizes. Symbol "x" denotes control measurements. Saw-tooth pattern in case of authentication algorithms is caused by alignment of data on a multiple integer of block size, which is 512 bits. In case of encryption algorithms the block size is smaller, usually 64 or 128 bits.

7. Model Verification Using Available Data from Other System Implementations

In this section, the model and method for throughput calculation for any packet size is verified using available data from other IPsec systems. This set of data represents certain variability between the system implementations.

In Tab. 3 – Tab. 8 are confronted measured (referenced) and calculated values of the throughput. Values that are input to the calculations are denoted with an asterisk. Besides presented examples, data from other implementations can be found in [26], [27], [28].

Verification has shown that proposed model and method are valid for a wide spectrum of security system implementations. The inaccuracy of calculation is keeping below 2 % in most cases.

7.1. Implementation 1

Intel corporation in [11] presents performance of cryptographic accelerator Intel EP80579 Integrated Processor with synchronous mode of operation. Operation system is Linux with IPsec implementation OpenS/WAN 2.4.9 and cryptographic framework Linux-OCF (port of native OpenBSD cryptographic framework to Linux). Security configuration is ESP-3des/AH-sha1. Comparison of referenced and calculated values is presented in Tab. 3.

7.2. Implementation 2

Authors in [24] tested performance of IPsec with encryption algorithm AES in GCM mode (Galois Counter Mode), which data first encrypts and then authenticates. Algorithm is based on AES-NI (AES New Instruction set), which was introduced in 2010 by Intel. System is without hardware accelerator, operation system is Linux with kernel 2.6.31, security configuration is ESP-aes_ni_gcm. Comparison of referenced and calculated values is presented in Tab. 4.

7.3. Implementation 3

Authors in [3] proposed a new IPsec implementation for processor Cavium Oceton CN58XX. They achieved throughput of 20 Gbps for packet size 1024 bytes what was in year 2010 the record-breaking achievement in academic and industrial research. Security configuration is ESP-aes128. Comparison of referenced and calculated values is presented in Tab 5.

7.4. Implementation 4

Authors in [23] tested performance of IPsec on a multi-core system using *pcrypt IPsec* for Linux, which utilizes the cores in parallel. Security configuration is ESP-aes192/ESP-sha1. Comparison of referenced and calculated values is presented in Tab. 6.

7.5. Implementation 5

Freescale in [4] tests performance of cryptographic accelerator PowerQUICC II MPC8360E with asynchronous mode of operation. Operating system is Linux with proprietary IPsec implementation *Mocana*. Security configuration is ESP-3des/ESP-sha1. Comparison of referenced and calculated values is presented in Tab. 7.

7.6. Implementation 6

As was mentioned in the beginning of the paper, analysis and principles presented for IPsec should hold analogically also for the other security protocols.

Author in [25] tested performance of securing using SSL protocol. Protocol is implemented as OpenSSL on operating system Linux with cryptographic framework OCF-Linux. The main processor is Intel Xcale 533 MHz, cryptographic accelerator is PCI Hifn7956 with synchronous mode of operation. Security configuration is SSL-aes128. Comparison of referenced and calculated values is presented in Tab. 8.

Tab.3: Comparison of measured (referenced) and calculated values of throughput for different packet sizes. Implementation 1. Calculated characteristic parameters: t_{fix} = 0,01357 ms, R_{alg} = 778 Mbps.

Packet size [bytes]	**Referenced throughput [Mbps]**	**Calculated throughput [Mbps]**	**Difference Δ**
64	37	37,5	1,3 %
128	74	74,7	0,9 %
256	148 *	-	-
512	291	290,5	0,2 %
1024	563	560,1	0,5 %
1400	746 *	-	-
Average difference			0,8 %

Tab.4: Comparison of measured (referenced) and calculated values of throughput for different packet sizes. Implementation 2. Calculated characteristic parameters: t_{fix} = 0,00517 ms, R_{alg} = 1993 Mbps.

Packet size [bytes]	**Referenced throughput [Mbps]**	**Calculated throughput [Mbps]**	**Difference Δ**
64	98*	-	-
128	184	184,6	0,3 %
256	332	336,8	1,4 %
512	552	572,9	3,7 %
768	745	747,5	0,3 %
1024	882*	-	-
1280	945	993	Limiting is bandwidth of network adapter
1454	953	1056	
Average difference			1,4 %

Tab.5: Comparison of measured (referenced) and calculated values of throughput for different packet sizes. Implementation 3. Calculated characteristic parameters: t_{fix} = 0,0000916 ms, R_{alg} = 25,8 Gbit·s^{-1}.

Packet size [bytes]	**Referenced throughput [Gbps]**	**Calculated throughput [Gbps]**	**Difference Δ**
64	4,5	4,54	0,8 %
128	7,6	7,72	1,5 %
256	11,9*	-	-
512	16,6	16,31	1,7 %
1024	20,0*	-	-
Average difference			1,3 %

Tab.6: Comparison of measured (referenced) and calculated values of throughput for different packet sizes. Implementation 4. Calculated characteristic parameters: t_{fix} = 0,00352 ms, R_{alg} = 2607 Mbps.

Packet size [bytes]	**Referenced throughput [Mbps]**	**Calculated throughput [Mbps]**	**Difference Δ**
46	100*	-	-
110	228	229,7	0,7 %
238	446*	-	-
1006	915	1214	Limiting is bandwidth of network adapter
1404	945	1430	

Tab.7: Comparison of measured (referenced) and calculated values of throughput for different packet sizes. Implementation 5. Calculated characteristic parameters: t_{fix} = 0,02076 ms, t_{oh} = 0,00826 ms, R_{alg} = 626 Mbps, L_{bp} = 915 bytes.

Packet size [bytes]	**Referenced throughput [Mbps]**	**Calculated throughput [Mbps]**	**Difference Δ**
64	24	24,7	2,9 %
128	49	49,6	1,1 %
256	99*	for calculation t_{fix}	
390	153	150,7	1,9 %
512	202	198,1	1,9 %
1024	382*	for calculation t_{oh} and R_{alg}	
1280	414	414,3	0,01 %
1456	432*	for calculation t_{oh} and R_{alg}	
Average difference		1,5 %	

Tab.8: Comparison of measured (referenced) and calculated values of throughput for different packet sizes. Implementation 6. Calculated characteristic parameters: t_{fix} = 0,4146 ms, R_{alg} = 23,517 Mbps.

Packet size [bytes]	**Referenced throughput [Mbps]**	**Calculated throughput [Mbps]**	**Difference Δ**
64	1,166*	-	-
256	3,992	4,061	1,6 %
1024	10,735	10,699	0,3 %
2048	14,707*	-	-
Average difference			0,9 %

8. Conclusion

The paper was focused on throughput of securing process, which cannot be described neither by a constant value of bits per second nor by a constant value of packets per second over the whole range of packet sizes. Based on analysis of hardware and software components of IPsec system a general throughput model of IPsec process was proposed. It comprises of characteristic parameters t_{fix} and R_{alg} for synchronous mode of operation and t_{fix}, t_{oh} and R_{alg} for the asynchronous mode of operation. These parameters are constant for any packet size.

A method for obtaining characteristic parameters was derived from the general throughput model. Usage of the method eliminates the need for performing many measurements for building the throughput profile over the whole range of packet sizes. The measurements can be rather time-consuming as one measurement can last several minutes using UDP iterative search technique. Characteristic parameters therefore might be practically used for comprehensive and convenient definition of securing process throughput on any security system.

Validation of the model was performed using data from various security system implementations – it provides results with an error below 2 % from the actual measured value in most cases.

The model and the method might be further used for a throughput calculation when more packet sizes are present in a mixed traffic and might be used as an input to a queuing model describing the security gateway where the service rate needs to be known.

Acknowledgements

This work is a part of research activities conducted at Slovak University of Technology Bratislava, Faculty of Electrical Engineering and Information Technology, Institute of Telecommunications, as a partial result of the Research & Development Operational Programme for the project Support of Center of Excellence for Smart Technologies, Systems and Services, ITMS 26240120029, co-funded by the ERDF.

References

[1] STALLINGS, William. *Network Security Essentials: Applications and Standards (4th Edition)*. New Jersey (USA): Prentice Hall, June 2011. ISBN 0-13-610805-9.

[2] KENT, Stephen and Karen SEO. *RFC 4301 - Security Architecture for the Internet Protocol* [online]. IETF RFC. December 2005. Available at: http://www.ietf.org/rfc/rfc4301.txt.

[3] MENG, Jinli. Towards high-performance IPsec on cavium OCTEON platform. In: *Proceedings of the Second international conference on Trusted Systems*. Berlin: Springer-Verlag, 2010, pp. 37–46. ISBN 978-3-642-25282-2.

[4] WATERS, Geoff. Understanding Crypto Performance in Embedded Systems. *EE Times Design* [online]. July 2009. Available at: http://www.eetimes.com/design/embedded/4008316/Understanding-Crypto-Performance-in-Embedded-Systems-Part-1.

[5] MINAAM, Diaa Salama Aabdul, Hatem M. ABDUAL-KADER and Mohiy Mohamed HADHOUD. Evaluating the Effects of Symmetric Cryptography Algorithms on Power Consumption for Different Data Types. *International Journal of Network Security*. 2010, vol. 11, no. 2, pp. 78-87. ISSN 1816-3548. Available at: http://ijns.femto.com.tw/contents/ijns-v11-n2/ijns-2010-v11-n2-p78-87.pdf.

[6] XENAKIS, Christos. A generic characterization of the overheads imposed by IPsec and associated cryptographic algorithms. *Computer Networks*. 2006, vol. 50, iss. 17, pp. 3225-3241. ISSN 1389-1286. DOI: 10.1016/j.comnet.2005.12.005.

[7] ROGAWSKI, Marcin and Jeans-Peter KAPS. Efficient Hardware Accelerator for IPSec Based on Partial Reconfiguration on Xilinx FPGAs. In: *Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference.* Cancun, Mexico: IEEE, December 2011, pp. 242–248. ISBN 978-0-7695-4551-6.

[8] MALIK, S. and R. MALHOTRA. Performance Enhancement for IPsec Processing on Multi-Core Systems. In: *Nassau Community College* [online]. June 2010. Available at: www.ncc.org.in/download.php?f=NCC2011/1569361787.pdf.

[9] SafeXcel IP: Flow Through Packet Engine. *Authentec* [online]. 2007. Available at: http://www.authentec.com/Portals/5/Documents/3113_SafeXcelIP_EIP-196_v02.pdf.

[10] STEVENS, Paul. Help Cut Network Security Cost. In: *Embedded Innovator* [online]. Fall 2008. Available at: http://download.intel.com/design/network/ica/downloads/EmbeddedInnovator100608.pdf.

[11] Accelerating a Security Appliance. In: *Intel* [online]. 320124-002US. February 2009. Available at: http://download.intel.com/design/intarch/ep80579/320124.pdf.

[12] BENVENUTI, Christian. *Understanding Linux Network Internals*. Sebastopol (USA): O'Reilly Media, 2005. ISBN 0596002556.

[13] KEROMYTIS, Angelos., Jason WRIGHT and Theo RAADT. The Design of the OpenBSD Cryptographic Framework. In: *Usenix Annual Technical Conference*. San Antonio, Texas (USA): USENIX, June 2003. ISBN 1-931971-11-0. Available at: http://www.openbsd.org/papers/ocf.pdf.

[14] PAEPS, Phillip. Crypto Acceleration on FreeBSD. In: *BSDCan - The Technical BSD Conference* [online]. Ottawa, 2010. Available at: http://2009.asiabsdcon.org/papers/abc2009-P1B-paper.pdf.

[15] CAVIUM. *NITROX Lite CN505 Security Macro Processor IPsec Product Brief.* 2005, http://www.caviumnetworks.com/pdfFiles/NITROX-XL_CN16XX-NFBE_PB.pdf.

[16] KABIR, Ahsan and Kim PHILLIPS. Cryptographic Asynchronicity and Linux. In *Embedded Linux Developers Conferrence* [online]. San Jose, 2007. Available at: http://www.mvista.com/download/presentations/Kabir_Phillips.pdf.

[17] FUN, L.S. Irene. *Layer 3 Forwarding* and *IPSec Measurement Methodology and Optimization* [online]. January 2009, no. 321068. Available at: ftp://download.intel.com/design/intarch/PAPERS/321068.pdf.

[18] Crypto Offload Options. In: *Elliptic* [online]. May 2008. Available at: http://www.ellipticsemi.com/pdf/whitepapers/Crypto_Acceleration_Options_81030.pdf.

[19] The Cisco QuantumFlow Processor: Cisco’s Next Generation Network Processor. In: *Cisco* [online]. Februar 2008. Available

at: http://www.cisco.com/en/US/prod/collateral/routers/ps9343/solution_overview_c22-448936.pdf.

[20] Manual: Packet Flow. In: *MikroTik* [online]. April 2010 [cit. 2012-09-11]. Available at: http://wiki.mikrotik.com/wiki/Manual:Packet_Flow.

[21] KENT, Stephen. RFC 4302 - IP Authentication Header. *IETF RFC* [online]. December 2005. Available at: http://tools.ietf.org/html/rfc4302.

[22] KENT, Stephen. RFC 4303 - IP Encapsulating Security Payload (ESP). *IETF RFC* [online]. December 2005. Available at: http://tools.ietf.org/html/rfc4303.

[23] KLASSERT, Steffen. Parallelizing IPsec: switching SMP to 'On' is not even half the way. In: *StrongSwan* [online]. 2010. Available at: www.strongswan.org/docs/Steffen_Klassert_Parallelizing_IPsec.pdf.

[24] HOBAN, Adrian. Using Intel® AES New Instructions and PCLMULQDQ to Significantly Improve IPSec Performance on Linux. In: *Intel Corporation* [online]. August 2010. Available at: http://download.intel.com/design/intarch/papers/324238.pdf.

[25] MCCULLOUGH, David. OCF-Linux. *Linux* [online]. Available at: http://ocf-linux.sourceforge.net.

[26] Comparing the Performance of Broadcom IPsec Boards. In: *BroadCom Corporation* [online]. 2002. Dostupné z: http://www.broadcom.com/collateral/wp/IPSEC-WP100-R.pdf.

[27] Performance of Hardware-accelerated IPsec VPN Systems. *EE Times Design* [online]. August 2007, Available at: http://www.eetimes.com/electrical-engineers/education-training/tech-papers/4136176/Performance-of-Hardware-accelerated-IPsec-VPN-Systems.

[28] SSL Benchmarks. *Mocana* [online]. Available at: https://www.mocana.com/benchmarks.html.

[29] VOZNAK, Miroslav and Filip REZAC. Web-based IP Telephony Penetration System Evaluating Level of Protection from Attacks. *WSEAS Transactions on Communications*. February 2011, vol. 10, iss. 2, pp. 66-76. ISSN 1109-2742.

About Authors

Adam TISOVSKY was born in Bratislava, Slovakia in February 1986. He received his master’s degree from Slovak University of Technology, Bratislava in 2009. Since 2009 he is a postgradual student at Institute of Telecommunications, Slovak University of Technology, Bratislava. He focuses on network security, cryptographic algorithms, QoS, VoIP and modeling the network parameters with respect for real-time applications.

Ivan BARONAK was born in Zilina, Slovakia in July 1955. He received the electronic engineering degree from Slovak Technical University Bratislava in 1980. Since 1981 he has been a lecturer at Institute of Telecommunications, STU Bratislava. Nowadays he works as a professor at Institute of Telecommunications of FEI STU Bratislava. Scientifically, professionally and pedagogically, he focuses on problems of digital switching systems, ATM, Telecommunication management (TMN), NGN, VoIP, QoS, problem of optimal modeling of private telecommunication networks and services.