Analysis and implementation of custom cipher algorithm for IPsec under Linux OS

Mladen Veinović[†], **Aleksandar Jevremović**[††] **and Goran Šimić**[†††],

Singidunum University, Serbia   Singidunum University, Serbia   Military Academy, Serbia

Summary
This paper deals with the issue of the protection of public network communications by using IPsec implementation in Linux kernel. Complete description of custom cipher algorithm implementation method is given. Paper also contains experiment description with results. In the existing literature there are numerous designs based on public ciphers depending on the specific product. It is known that professional security systems require their own design, cipher and complete control over cryptographic keys. Due to the complexity of lower level protocols and kernel modification, the most frequently implemented solutions are those of application level. The advantages of implementation level (IP) used in this work - better performance, higher level stability and end users relaxation - are presented in paper. It is very important to point out that the focus of this paper is not better performance achievement, but finding the adequate basis for application of one's own (custom) algorithm.

Key words:
Custom cipher algorithm, IPsec, Linux kernel, network security.

1. Introduction

Cryptographical mechanisms represent the basis of professional data protection systems. Nowadays, they represent the most frequently used way in the protection of open computer networks. There are many systems designed for protecting different kinds of data interchange: secured file transfer, secured e-mail service and e-payment systems are just some of them. Although, there are many commercialized cryptographically techniques, the end users are never satisfied and they have more and more requirements. The protection systems with traditionally symmetric techniques show the best characteristics. Unfortunately, the authentication handling and transaction integrity requires using asymmetric cryptography systems with public key infrastructure (PKI).
The computer networks security level depends on correct implementation and simplicity of using (installing and exploiting) of concrete solution. According to this fact, the transparent protection system represents the best solution for end business users. The protection, implemented at the lower layer of computer protocols, enables the continuous using of business applications by the end users. This way, they don't have to follow the strict security requirements in which they can make mistakes. Beside this, in many cases it is impossible to change existing applications and to extend them with additional protection mechanisms. The standardization specifications at the lower levels are more narrowed and less changeable than on the higher OSI levels. Therefore, the lower level protection solutions are less platform dependant than solutions at the higher OSI levels.
The security is guaranteed in the professional protection systems (government, military...) by using custom algorithms in symmetric encrypting systems and private symmetric encrypting keys. The using of reliable software components represents the additional guarantee. The open source components represent some of them. They are compiled during installation time, according to the hardware and software platform. The open source components provide to developers the full control of the protection system executing. This way they can make reliable software evaluation.
The solution of secure data transfer over unsecured communication channels (such as Internet infrastructure), is represented in this paper. This solution is implemented at IP level and on Linux OS platform. The project has focused on three preconditions:

1. Transparency and easy to use for end users,
2. Custom encrypting algorithm,
3. Reliable open source components.

The IP level protection [17], based on Linux platform and using custom algorithm, offers the satisfied performance. The complete analysis of this solution can be performed through long time tests, and in different network environments. The next section describes the actual solutions for network communications security. The proposed solution is described in the 3rd section.

2. IPsec framework

IPsec (IP Security) represents one of the most popular network level encrypting solutions. IPsec also represents the IP (Internet Protocol) security standard. It consists of encrypting protocol set which is designed for packet transfer and key interchange [16]: ESP (Encapsulating Security Payload – authentication, privacy, and message integrity), AH (Authentication Header – authentication and message integrity, without privacy), and IKE (Internet Key

Manuscript received July 5, 2008.
Manuscript revised July 20, 2008.

Exchange).
IPsec represents the service, implemented on top of the network OSI layer. This means that all client's applications implemented on the upper layers can be served by IPsec. This is a great advantage with regard to the very popular SSL or TLS security protocols which are designed for serving the application layer clients. IPsec can also protect TCP and UDP based protocols (Fig. 1).

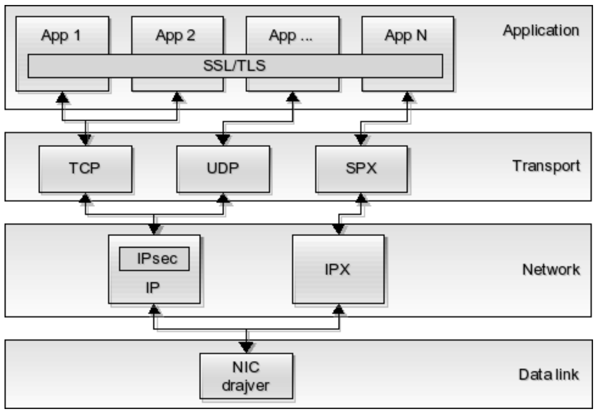

Fig.1. Different protection levels (TCP/IP stack) and solutions

There are many IPsec implementations: FreeS/WAN [2], OpenS/WAN [3], StrongS/WAN [4]. These applications depend on different OS: MS Windows, FreeBSD, NetBSD, OpenBSD, Mac OS X, AIX, Cisco, and Linux. Our security solution is developed on IPsec implementation, which is integrated in Linux OS kernel (Alexey Kuznetsov and David S. Miller [5]). This IPsec implementation has been regular part of the Linux v.2.6 kernel (and later) since 2002. The main advantages of this implementation are:

- Source code accessibility,
- Kernel integration provides high performance,
- Possibility of adding extensions (such as encrypting algorithm),
- License (General Public Licence – GPL).

3. Custom solution proposal

This hypotetical case stydy includes two users. They communicate from two different local networks over the Internet (unsecured public network). The solution is implemented at the network (IP) layer of the OSI model. This produces low coupling with the environmental components and independence of client applications. Solution approach is motivated by these two characteristics. As mentioned above, the transparency and simplicity represent other benefits. The complex implementation and high cost are the main disadvantages of lower level solutions.
The decision about the network point at which the solution should be implemented represents another problem. If the local network (LAN) is considered to be secure (data can be propagated in the plain format inside the LAN), the gateway interfaces represent the best places for the solution implementation. The gateways also represent the border between the secured (LAN) and unsecured (Internet) network (Fig. 2).

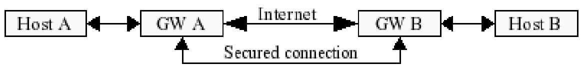

Fig.2. Secured network points (gateways)

By implementing solution on the gateway, the protection components have to be installed just on the gateway platform (computer). This way, the number of devices used by solution is minimized. The additional advantage is that the administrator can decide in which cases the encrypting will be used. The system can allow the open data traffic, such as encrypted traffic. The information header data can be used by system for deciding if the information will be encrypted or not. This approach provides the great flexibility of the system, without endangering the existing communications. The secure policy management is simplified by administering the minimized number of computers.

4. Using of custom encrypting algorithm

The symmetric encrypting system with the algorithm which generates pseudo random arrays represents the standard solution for the data protection. The symmetric means that this algorithm is initialized by secret symmetric keys. The pseudo random array generators are used in this process. Their synchronization is based on the key interchange between them. They also interchange some additional parameters. The data protection is based on secrecy and quality of the symmetric encrypting keys. Therefore, the design of algorithm has to protect the system from the cryptographically attacks directed to it.
The products available on the market consist of different standard algorithms such as 3DES [6] and AES [7]. These algorithms use predefined size of the secret keys, and this size is less or equal to 256 bits. Often, users can choose the encryption algorithm from the client application. Existing standard algorithms are mainly designed as separate modules. The concrete algorithm is specified during the cryptology synchronization process.
Security reasons motivate the clients to develop and implement custom algorithm, different than the others on the market. For reliability reasons, the compatibility between the algorithm and the specific (trusted) hardware and software platform is also required. The rate performance represents another important requirement. The algorithm execution time has to be as short as possible. Therefore, the algorithm of this solution (protection at the IP layer of OSI model) is implemented in the operating

system (OS) kernel.

5. Reliability of solution components

The reliability of complete solution depends on reliability of each component. The evaluation of the overall solution means that the reliability of each component has to be evaluated. The accessibility of the each component source code represents the one of the most important evaluation preconditions. The OS represented the platform of this solution. Therefore it is considered as the base component of the protection system.

Linux OS represents one of the most appropriate base components for data protecting. There are several reasons for this statement:

- Source code availability,
- High networking possibilities and performances,
- Kernel integrated firewall,
- Kernel integrated IP security,
- Developing of custom encrypting algorithm as a kernel module.

There are many security products based on Linux OS. These products are mainly designed for LANs protection based on firewall concept (SmoothWall [8], IPCop [9], Trustix [10]). Some Linux producers present their software as the integrated security solutions (Innominate AG [11], secunet Security Networks AG [12], etc.). The mentioned companies prefer Linux as the protection platform better than MS Windows because they can access to overall Linux source code and they can adopt this source to the special protection needs.

6. Model performances analysis

The presented solution is implemented at gateway point. The gateway represents the most overloaded LAN point. All of the data interchange between LAN and environmental network pass through the gateway. Therefore, the clients expect the best performance of gateway software. It must run as fast as possible. Protecting solution must be able to encrypt whole amount of data interchanged through the gateway. The maximum traffic density which should be processed can be determined by lower capacity network link (if the gateway is connected to two network links).

The exact ratio between process power and bit rate which has to be protected, can be determined by concrete algorithm analysis and by testing with the realistic (expected) data traffic. Regarding the high network performances of Linux OS, and the benefits of the firewall and IP security implementation in the OS kernel, the algorithm represents the most critical point of the protection process. On the other hand, the complex algorithms which are required in the professional systems are great time consumers.

The scalability represents other measure of quality of the concrete solution. The scalability exactly means that the system performance can be improved by using additional hardware (e.g. fast signal processors, programmable hardware – FPGA and similar technologies). The secure encrypting and decrypting functions and relaxing of gateway's CPU time represent the main advantages of this solution. The random array generators, encrypting key place holders, and smart card drivers can be sited in additional hardware modules. These modules increase the solution security and other performance.

7. Solution model

The solution algorithm is realized as the kernel module [17]. It is implemented in C language by using standard Linux kernel Crypto API [13] (Fig.3).

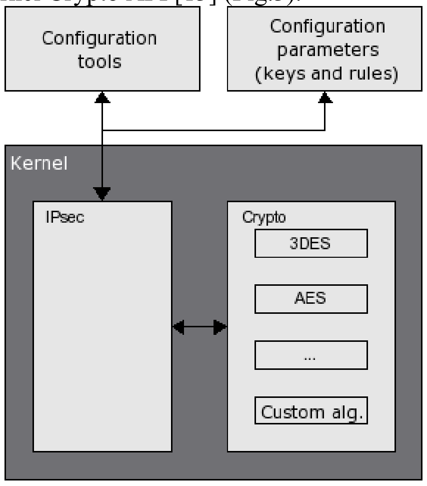

Fig.3. Solution model

The basic kernel Crypto API functions are:

- algorithmname_setkey,
- algorithmname_encrypt,
- algorithmname_decrypt.

Additionaly, there is the key manipulating structure. For testing and evaluating purposes, the special algorithm named mAES is developed. mAES parameters (the number of additional structures, complexity and the simetric secret key length) are adapted to be greater than in the comercial AES algorithm. Although the average time (measured during simulation) of pseudo random array generating in mAES algorithm, is two times longer than one in the AES algorithm the main focus is on integration of the new algorithm (with better protecting performances) in the OS kernel.

mAES algorithm is designed for testing purposes only. AES algorithm represents the base for mAES algorithm with modification of encryption/decryption functions. Although these modifications have negative influence for

performances, the main purpose of the custom algorithm developement was to examine implementation process.
There are two steps to include the new (mAES) algorithm into the kernel. Firstly, some kernel configuration parameters have to be changed:

- CONFIG_INET_AH=y/m
  specification of authentication header
- CONFIG_INET_ESP=y/m
  specification of encapsulating security payload
- CONFIG_INET_IPCOMP=y/m
  specification of support of Payload Compression Protocol (IPComp – RFC3173)
- CONFIG_CRYPTO_mAES=y/m
  specification of custom algorithm

Statement option y/m defines the way in which some kernel functions will be implemented by including the IPsec support (m module) or the kernel part designed for protection (y module).
The other step is compiling the kernel source code. Then, the OS kernel with IPsec support and with custom encripting algorithm is ready for use.
Besides the kernel preparation, some additional software packages have to be included: the kernel conrol package and the IPsec conrol package. Kame Project IPsec-Tools [14] is the one of the most frequently used package. This package is specialy designed for Linux OS kernel integrated IPsec manipulating. IPsec-Tools consists of four basic components [18]:

- libipsec – PF_KEY implementation library
- setkey – manipulation tool for kernel's SPD (Security Policy Database) and kernel's SAD (Security Association Database)
- racoon – Internet Key Exchange (IKE) daemon designed for key management in IPsec connections.
- racoonctl – racoon daemon control tool

ISAKMP/Oakley [15] represents the other package ported from OpenBSD systems.
The Linux distribution with mAES IPsec solution would include all packages related to kernel and Ipsec, such as additilonal software packages mentioned above. The developement of the separate Linux distribution represents the hard job. Therefore, the using of an existing distribution as a platform, represents the better solution. This distribution has to have high stability, high networking performance and to be standardized. Slackware represents the most appropriate Linux distribution according to these criteria [1].
The configuration tools are designed for working rules configuration (specifying the encrypting keys according to specific usage cases).

8. Security solution analysis

The networked system configuration, considered in this paper, is in permanent contact with the external users from unsecure network (Internet). Therefore, the system security represents very important parameter. The potential attacks may produce the access to the system administration functions, system breakdown, key robbery, key changing, key deleting, changing of data encrypting policy, access to local resources, etc.
The system security is based on Linux OS reliability and IPsec implementation such as kernel configuration and using of additional software packages. The accessibility of the Linux OS source code represents the main security risk. Potential attacker can analyse Linux and IPsec implementation. It can find bugs and use them to compromise the system stability, functionality and make unauthorized access to the system resources.
This problem can be solved by:

- Excluding unnecessary kernel parts (modules),
- Disabling unnecessary functionalities,
- Appropriate and tested system configuration,
- Updating mandatory components according to provider's recomendations,
- Using IPS/IDS (Intrusion Prevention System / Intrusion Detection System) system and other systems which guarantee impossibility of system modification without authority cheking,
- Keeping event logs about sistem executing and attacking atempts on the separate network node.

Another problem is using standard cipher algorithms (such as AES and DES). If there are mistakes in these algorithms, than potential attackers can record the traffic performed by using these algrithms, and use mistakes to decript the trafic content. In this situation traffic can be considered as an open form of data interchange, and the unauthorized access to the gateway is not neccesary. Using custom algorithm represents the solution for this serious problem.

9. Experiment description

The three networked computers are used in experiment. The two of them are used as the communication end point with the implemented IPsec solution. The third one is used as the monitoring (medium) point which records the whole traffic between two end points (Fig.4).

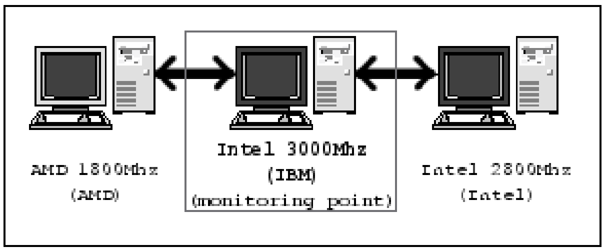

Fig. 4. Scheme of computer network used in experiment

The computers are named by acronyms (AMD, Intel, IBM). The properties of computers, which are important for the experiment, are shown in the next review (table 1).

Table 1: Hardware characteristics of computers used in experiment

AMD	
Processor:	AMD Sempron(TM) 2200+ 1800MHz 256KB cache
RAM:	256MB
Network Adapter:	Realtek RTL-8169 Gigabit Ethernet
Intel	
Processor:	Intel(R) Pentium(R) D 2.8GHz 1024KB cache
RAM:	4096MB
Network Adapter:	NetLink BCM5789 Gigabit Ethernet
IBM	
Processor:	2 x Intel(R) Xeon(TM) CPU 2 x 3.00GHz 2 x 2048KB cache
RAM:	4096MB
Network Adapter:	2 x NetXtreme BCM5721 Gigabit Ethernet

As mentioned above, AMD and Intel computers have implemented IPsec solution. IBM computer records the communication between AMD and Intel and checks the encription results.

The Slackware Linux 12 with kernel 2.6.21.5. is installed on each computer. IPsec is enabled on AMD and Intel computers:

- CONFIG_NET_KEY = Y
- CONFIG_INET_AH = Y
- CONFIG_INET_ESP = Y

The algorithm setup is configured as folowing:

- CONFIG_CRYPTO_HMAC = Y
- CONFIG_CRYPTO_MD5 = Y
- CONFIG_CRYPTO_SHA1 = Y
- CONFIG_CRYPTO_DES = Y
- CONFIG_CRYPTO_AES = Y
- CONFIG_CRYPTO_ mAES = Y

IBM computer bridge module is enabled in order to perform midpoint role:

- CONFIG_BRIDGE = Y

As additional software, Netperf program is activated at AMD and Intel computers and Ethereal program is activated at IBM computer.

Netperf program provides the execution of different network tests. This program consists of client and server components. This way, it is able to monitor different client and server side parameters at the same time. The test results are collected by using Netperf.

Ethereal program is designed for recording the network traffic which passes through the host platform. It simplifies the analysing by layering the traffic regarding the protocol levels. This program is used in the experiment, for controling the traffic between AMD and Intel computers and for data encrypting control (to ensure that encripting engine work properly).

The main caracteristics of the experiment are:

- TCP stream (10 seconds duration) generated by Netperf program is used
- The measuring is performed three times for each algorithm (AES, 3DES and mAES) as open communication (without using of IPsec)
- Each measuring is performed two times in two directions (from AMD to Intel and vice versa)
- AH (Authentication Header) and ESP (Encapsulated Security Payload) are used in each communication
- The same hardware and networking platform is used for each measuring.

The measured parameters are:

- Dataflow rate,
- Sender CPU load,
- Receiver CPU load.

The same keys are used in each algorithm (128 bit for AH and 192 bit for ESP).

10. Experiment results analysis

The experiment results are as expected. The next ilustration (Fig.5) shows the dataflow rate by using different encrypting algorithms.

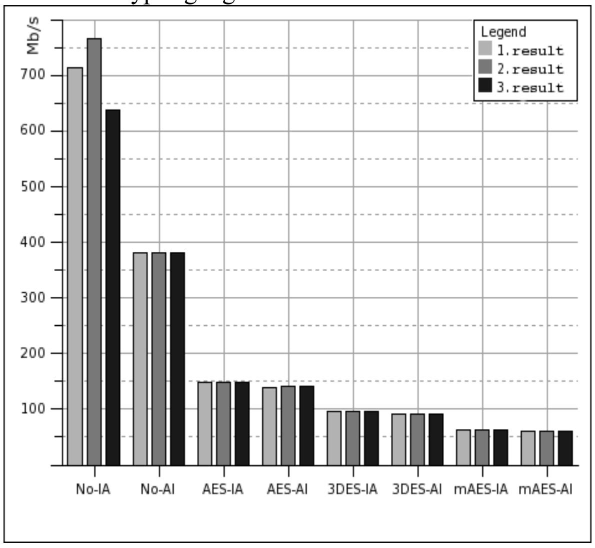

Fig.5. Dataflow rate vs. using different algorithms

The first two bar groups (No-IA, No-AI) represent the dataflow rate without encription. These groups are important because they point up the significant difference between the CPU performances. The Intel CPU sending rate (No-IA) is almost two times better than AMD one (No-AI).
This difference is overcome by using the encryption algorithms. In this case, the dataflow rate follows the algorithm complexity. The same hardware platform and communication chanel is used in each measuring. This means the CPU load is the main reason of the rate lagging. The performance differenes betwen separate algorithms fluctuate from 30% to 60%.

Table 2: The results of measuring

Dataflow rate (Mb/s)	**Sender CPU load(%)**	**Receiver CPU load(%)**
No enc. (Intel -> AMD)		
712.41	14.52	77.52
765.25	15.06	84.31
636.59	12.73	69.63
No enc. (AMD -> Intel)		
381.11	42.56	22.38
380.91	41.86	21.33
380.96	42.17	22.34
AES (Intel -> AMD)		
147.99	36.22	99.90
148.15	36.18	99.90
148.03	36.25	99.90
AES (AMD -> Intel)		
140.05	99.90	47.93
140.47	99.90	47.89
140.25	99.90	47.82
3DES (Intel -> AMD)		
96.25	35.93	99.90
95.70	35.82	99.90
95.51	35.92	99.90
3DES (AMD -> Intel)		
91.68	99.90	40.99
91.55	99.90	40.01
91.57	99.90	41.23
mAES (Intel -> AMD)		
63.72	30.83	99.90
63.75	31.00	99.90
63.74	30.89	99.90
mAES (AMD -> Intel)		
61.33	99.90	37.22
61.44	99.90	36.28
61.40	99.90	36.07

The measured numerical values are represented in the table above (table 2). AMD CPU is overloaded in each case (99,9%) regardless the role in communication (sender or receiver). This fact points up that the processing power represents the critical resource in encryption. Also, it is possible to predict the increasing of dataflow rate by using more powerfull CPU.
Note that the experiment is not focused to compare the processor performances. The AMD CPU is much older (it runs with less cache memory and on lower frequency) than Intel CPU, and worse performace is expected. The purpose of experiment was to evaluate loading on different platforms caused by encripting great amount of data using different algorithms.

11. Conclusions

The solution described in this paper is related to the problem of confidental data transfer over the unsecure public network by using custom encryption algorithm. The solution is developed by using OpenSource technologies. This way it is possible to evaluate the reliability of each component. Process of custom algorithm implementation by using C programming language and Linux kernel Crypto API is described and analysed. The proposed solution is based on using standard Crypto API functions. Slackware Linux distribution is used as the OS base.
The proposed solution enables implementation of security services in computer networks without end users overloading. Hiding solution from the end users also eliminates the need for extra training and eliminates possibilities of errors at their level.
The different aspects of IP level data protection are considered. Complete process of custom algorithm implementation in Linux kernel is described.
One of the conclusions is the today's x86 computer's are inadequate for data encryption and for maximum utilization of network channel's throughput. The experiment results show that additional hardware should be used (instead of software algorithm implementation) in order to improve the solution performance.
mAES solution is focused on unsecure network channels (e.g.Internet). The throughput of these channels is usualy much lower than gigabit LAN channels. Therefore, the utilization in this case should include the data compression before their encryption.
Further research could be focused on secret keys management (esspecialy keys distribution and storage on SmartCards), algorithm optimization and solution scalability (usage of additional hardware resources).

References

[1] Daniel de Kok, "Securing IP traffic with IPsec" The Slack World #1, 2005.
[2] Claudia Schmeing, "FreeS/WAN documentation" [Online]. Available: http://www.freeswan.org/
[3] "OpenSWAN documentation" [Online]. Available: http://www.openswan.org/
[4] Andreas Steffen, "strongSwan documentation" [Online].

Available: http://www.strongswan.org/
[5] Ken Bantoft, "The Future of IPsec on Linux" [Online]. Available: www.xelerance.com/talks/linuxtag2004/IPseconLinux.pdf
[6] National Institute of Standards and Technology, "Data Encryption Standard (DES)" [Online]. Available: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf
[7] Journal of Research of the National Institute of Standards and Technology, "NIST reports measurable success of Advanced Encryption Standard", Volume May-June, 2002
[8] [Online]. Available: http://www.smoothwall.org/
[9] [Online]. Available: http://ipcop.org/
[10] [Online]. Available: http://www.trustix.org/
[11] [Online]. Available: http://www.innominate.de/
[12] [Online]. Available: http://www.secunet.ch/
[13] The GNU/Linux CryptoAPI [Online]. Available: http://www.kerneli.org
[14] Qing Li, Jinmei Tatuya, Keiichi Shima, "IPv6 Advanced Protocols Implementation", 2007
[15] Network Working Group, "The Internet IP Security Domain of Interpretation for ISAKMP", RFC2407
[16] Donald Eastlake 3rd "Cryptographic Algorithm Implementation Requirements For ESP And AH",, 2004.
[17] Stephen Kent, Karen Seo, "Security Architecture for the Internet Protocol", 2005.

Dr Mladen Veinović, professor at Singidunum University, Danijelova 32 Belgrade, Serbia

Mr Aleksandar Jevremović, professor assistant at Singidunum University, Danijelova 32 Belgrade, Serbia

Mr Goran Šimić, professor assistant at Military Academy, Pavla Jurišića Šturma 33, Belgrade,