# Software Architecture for IPSec Crypto Offload Based on Security Processor

Xiaotie Qin*

*Library, Chongqing University of Science and Technology, Chongqing, 401331, China*

**Abstract:** This paper proposes efficient software architecture for ipsec crypto offload. It describes how to implement the control plane part and data plane part, analyzes each module in the data plane. Meanwhile, it provides details performance analysis.

**Keywords**: IPSec, offload, octeon, security association, security policy.

## 1. INTRODUCTION

With the rapid development of the Internet and E-commerce, network security problems have been attached more importance, which has led to the emergence of various safety communication protocols, such as WEP, IPSEC, SSL and so on [1]. Applied at all levels of the network protocol stack, these protocols feature identity authentication, encryption, integrity detection, and replay protection, etc. for the network flow. As the basis of security protocols, encryption algorithm is essentially a computationally intensive operation and demands larger processing capacity of CPU. On the other hand, there are some limitations on running IPSEC equipment such as limited processing speed and memory, etc. The data flow which increases rapidly and suddenly will consume large resources of the equipment [2], and consequently the equipment will run slowly, unable to process other tasks. How to eliminate the bad effects is what we need to take into account.

In the embedded system, we often use encryption accelerators and security processors, which can help improve safety and meanwhile have no significant impact on the overall performance and power consumption of the system. However, in order to achieve the predetermined high performance, we need to design efficient software architecture to realize encryption load-sharing-to off load encryption operations from host processors. Encryption load-sharing refers to separating the encryption operation which involves massive computation and consumes much computing time of CPU from the host processor and deliver it to the special encryption processor or security management [3]. As is known to all, in the communication from the client to the server, IPSec is often selected to facilitate the safe communication from end to end. In this paper, we have designed for IPSec crypto offload an efficient software architecture, which is based on a most advanced OCTEON CN63XX multi-core processor in the world [4-6].

In this paper, we present and fulfill an efficient software architecture used for the encryption load-sharing of the IPSec protocol–to off load encryption operations, and assess its performance based on a most advanced multi-core processor in the world.

## 2. INTRODUCTION TO IPSEC

IPSec is a set of protocol family to provide security mechanism for the IP level. The realization of IPSec requires two data bases, that is, Security Policy Data Base (SPDB) and Security Association Data Base (SADB). These databases determine the flow between the two IPSec Peers as well as what encryption algorithm and mechanism can be used to ensure safety. Besides, the IPSec system also includes components such as IKE (Internet Key Exchange), management of configuration application and transcoding methods, etc. I general, in the process of concrete realization, the following components will be fulfilled:

(1) The basic protocols of IPSec: used to process packet headers and interact with SPDB and SADB in order to determine what security control can be provided for data package and solve the network layer problems like segmentation and PMTU [7-9].

(2) SPDB: plays a decisive role in the safety control adopted by the data package. The processing of both the outcoming packet and incoming packet needs to consult the SPDB. For the outcoming packet, IPSec judges whether this packet needs safety control by retrieving SPDB. For the incoming packet, IPSec judges whether the safety control for this packet is in conformity with the safety control for policy configuration.

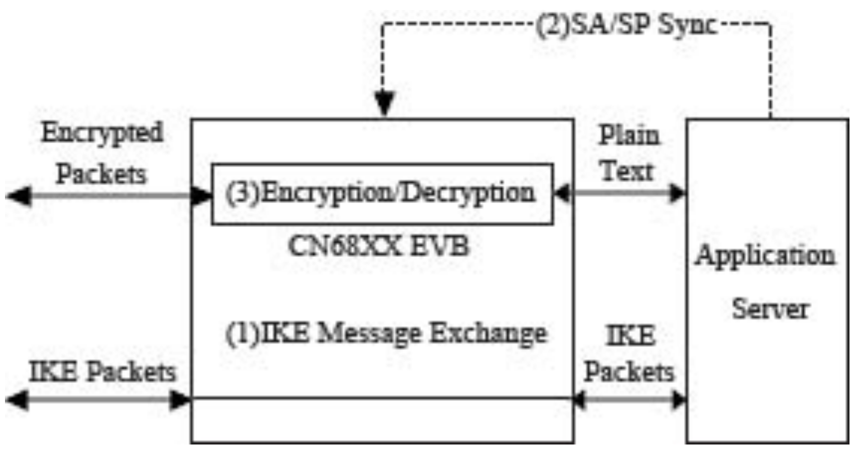

**Fig. (1).** Packets switching figure.

(3) SADB: for the processing of the outcoming packet and incoming packet, to maintain a SA list of an activity. The outcming SA is used to ensure the safety of the outcoming packet, while the incoming SA is used to process the incoming packet with IPSec heads [10].

(4) IKE: generally speaking, IKE is a user-level processing, except in the embedded operation system. IKE is responsible for creating the SA between the nodes of safe communication.

(5) Management of SP and SA: refers to the application which manages policy and SA.

IPSec includes two different protocols: ESP and AH. ESP will encrypt and verify the load of IP packet, yet the verification operation is not compulsory but optional. During the transmission process, some fields may be modified, while all the other fields except these ones will be verified by AH. To enable verification, ESP and AH will do computation on the relevant fields of the data package, thus getting a cryptographic HMAC (Harshed Message Authentication Code). For the outcoming data package, add the acquired HMAC to the data package; for the incoming data package, compare and verify the acquired HMAC with the build-in HMAC in the data package. Both ESP and AH have two package modes: Tunnel Mode and Transport Mode. Which package mode is to be chosen depends on whether to encrypt the whole IP packet or to encrypt only the load of the IP packet.

## 3. INTRODUCTION TO THE SYSTEM FRAMEWORK

IPSec crypto offload separates the compute-intensive encryption and decryption, thus relieving the pressure of application server. On the other hand, the encryption coprocessor of the OCTEON processor can accelerate the encryption and decryption of data. IPSec enables communication security from end to end. To encrypt and decrypt data, IPSec needs to pre-configurate security algorithm and secret key, which is called SA (Security Association). For the outcoming data package, an extra data structure needs to be configurated, which is known as SP (Security Policy). SP is used to check whether the outcoming data package needs encryption and verification. There are two ways to establish SA and SP: one is automatic establishment *via* IKE, another *via* the manual configuration of the system administrator. In this paper, the establishment of SA and SP is accomplished on the application server, while the processing of IPSec packet is accomplished by OCTEON processor, which requires the synchronization of SADB and SPDB between the OCTEON processor and the application server.

As is shown in the above Fig. (**1**), IPSec crypto offload can transmit IKE message to the application server according to the scheduling algorithm of the configuration:

(1) For the subsequent communication need, establish SA between the application server and the remote client. Run a listener program by the name of "listen-netlink" on the application server, and thus the listener program can monitor SA after its establishment between the application server and the remote client. And then send the monitored SA message to the OCTEON processor. At the same time, run a receiving program by the name of "ipsec_offlaod_recv_nl_msg", which can receive the SA/SP message from the application server.

(2) IPSec crypto offload uses the received SA/SP message to encrypt the plaintext data package sent from the application server or to decrypt the ciphertext data package sent from the remote client to the application server.

(3) What is relatively special is that for the incoming ciphertext data package, after IPSec crypto offload finishes decryption, the plaintext load will still be put in the ciphertext; moreover ESP/AH head and field will not be deleted, and then transmit the decrypted data package with ESP/AH head and field to the application server. Thus, the application server, *via* SPI in the ESP/AH head, can still retrieve SA, update the relevant field in SADB and keep synchronization with SADB in the OCTEON processor.

Besides, we need to put a patch on the Linux core of the application server, and make the application server possess the following functions:

(1) When the application server sends out a data package, which is applied to SA/SP, we should make the application server skip the step of encrypting the data, because all the data packages sent by the application server should be encrypted by IPSec crypto offload. Therefore, the application server still sends out the plaintext package.

(2) When the application server receives a data package, which is applied to SA/SP, the application server should skip the step of encrypting the data, because the data package has been decrypted by IPSec crypto offload.

(3) In the above passages, we've mentioned that after IPSec crypto offload decrypts the data package of the application server, ESP/AH head and field will not be deleted but will still be transmitted to the application server, which can update SA by using the relevant field in ESP/AH head, so we can maintain SADB and keep synchronization with SADB in the OCTEON processor.

(4) The application server is also responsible for sending massages about SA/SP delete events or update event to the OCTEON processor.

## 4. SOFTWARE ARCHITECTURE

The whole IPSec crypto offload software is composed of two parts: control platform and platform and data platform. Run an embedded Linux image in the control platform (usu. Core 0 as the control platform); dataplatform runs the Simple Executive of OCTEON, i.e. our IPSec crypto offload software.

OCTEON management has two types of NI (network port): one is the management interface in charge of information transmission; another is the Quintillion Ethernet interface in charge of data flow transmission. MontaVista Linux, a released version of embedded Linux will be uploaded in the first Core of OCTEON, used as the control platform of IPSec crypto offload. It will be responsible for managing and running application programs and other management functions of the control platform. The application programs of the control platform provide users with functions such as configuring SP, acquiring the running state and recording log of the program, etc.

The above figure shows the software framework of IPSec crypto offload. The data package flow between the application server and the remote client will enter the OCTEON processor. The Input Manager Module in the control platform will first process the data package which has entered the OCTEON processor. For example it will first compare the 5-tuple flow information of the data package with the Sa/SP message. If the data package is encrypted and applies to SA/SP, it should be decrypted, and so on.

If the entered data package is IP fragmentor, the restructuring function of IP fragmentator needs to be used to reassemble IP fragmentator. If the reassembled data package is from the decryption port and its protocol type is ESP/AH, it will be transmitted to the Decrypt Module. If the reassembled data package is from the encryption port, it will be transmitted to the Flow Manager Module, in which a session will be established based on the 5-tuple flow information, and message and SP related to the session will be extracted from SPDB. In the Encrypt Module, according to the SP message in SPDB, encrypt the plaintext data package. For the first data package in the session, the system will look up in SPDB established in the control platform according to the 5-tuple flow information of the data package. SP, which has been found, is used to encrypt the data package. In the end, the encrypted /decrypted data will be transmitted to the Output Manager Module. If the encrypted data package is longer than MTU, it will be sent after it is fragmented by the IP Fragmentor Module.

SPDB is used by the data platform, but formed by the configuraition of the control platform. The control platform runs an application program named ipsec_cryptor_offload_app_recv_nl_msg to receive SA/SP message from the application server, in which a listener program named ip is run. The application program in the control platform is responsible for writing the received SA/SP message in SADB and SPDB. The data platform will never modify SPDB, but will look it up when needed. However, for SADB, the data platform, besides looking it up, will most probably modify the Sequence Number of the secret key.

1. The Software Framework of the Control Platform

In the control platform, there is a network port named mgmt0, which is especially used by the embedded Linux of the control platform. This port is independent of the data platform and is linked to the OCTEON processor via PIC slot. In the core of the embedded Linux of the control platform is uploaded the Intel e1000 drive, through which this network port can work normally.

As is shown in the left half of Fig. (**2**), there are two types of drives of the network port. The application program run in the control platform must at least support the following two types:

Application type 1: the application program run in the control platform can process the data package received at the management port. This application program receives the data package from the management port, and sends it after processing it via the management port. The standard TCP/IP protocol stack can be used directly in the control platform in order to process Telnet or ordinary application programs based on socket.

Application type 2: the 2$^{nd}$ type of application program run in the control platform offers CLI function and is used to count record and store the relevant update. This kind of application programs visit the assigned shared memory block. Users can visit various shared memory blocks to update SADB/SPDB and extract statistics and logs, etc.

2. The Software Framework of the Data Platform

The data platform of IPSec crypto offload assumes functions such as block, transmission, encrypting/decrypting

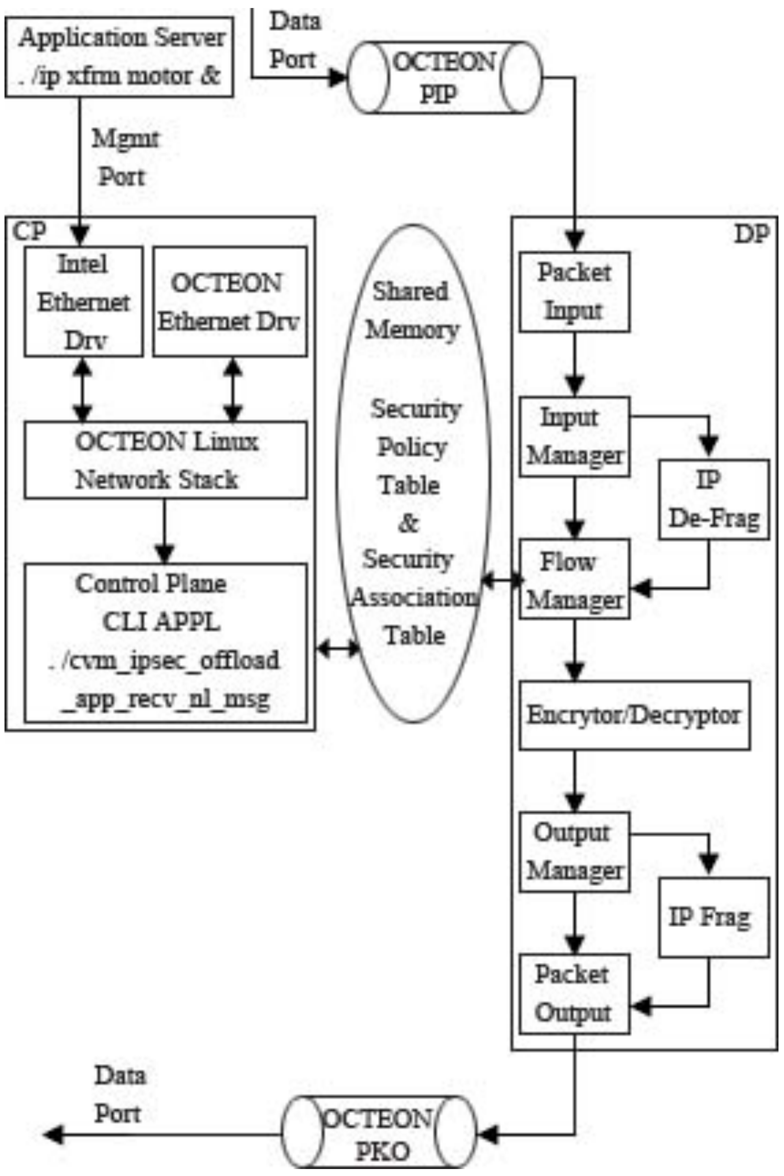

**Fig. (2).** Control platform and data platform.

the data package, etc. the whole data platform is composed of the following modules: they are Input Management Module, Output Management Module, Fragmentor Module, Reconfiguration Module, Flow Cotrol Module, SA-DB/SPDB Management Module, Encryption Module and Decryption Module. All the functions can be fulfilled in one core. The flow processed by different cores comes from different streams and sessions. The following figure shows the relationship between different modules.

The data package input: to process the data package entering the OCTEON platform, the hardware will assign a Group IG to the data package according to the data flow or the session. Thus, the data package with different group numbers will be processed by different cores in the data platform. In this way, the data package belonging to the same session will be processed by the same core.

The input management module: for the incoming data package, the input management module will first verify it. After being verified, if the data packages enter from the decryption port and also they are ESP/AH protocol packages, the data will be transmitted to the decryption module. Other data packages will be transmitted to the flow control module for further processing. If the data packages are fragmentors, before transmitted to the flow control module, they will be reconfigurated and integrated by the reconfiguration module.

The reconfiguration module: except the first fragmentor of the data package, the IP fragmentor does not include the 4th-level packet header. Therefore, these fragmentors need to be processed by the reconfiguration module in order to make SP applicable to these data packages.

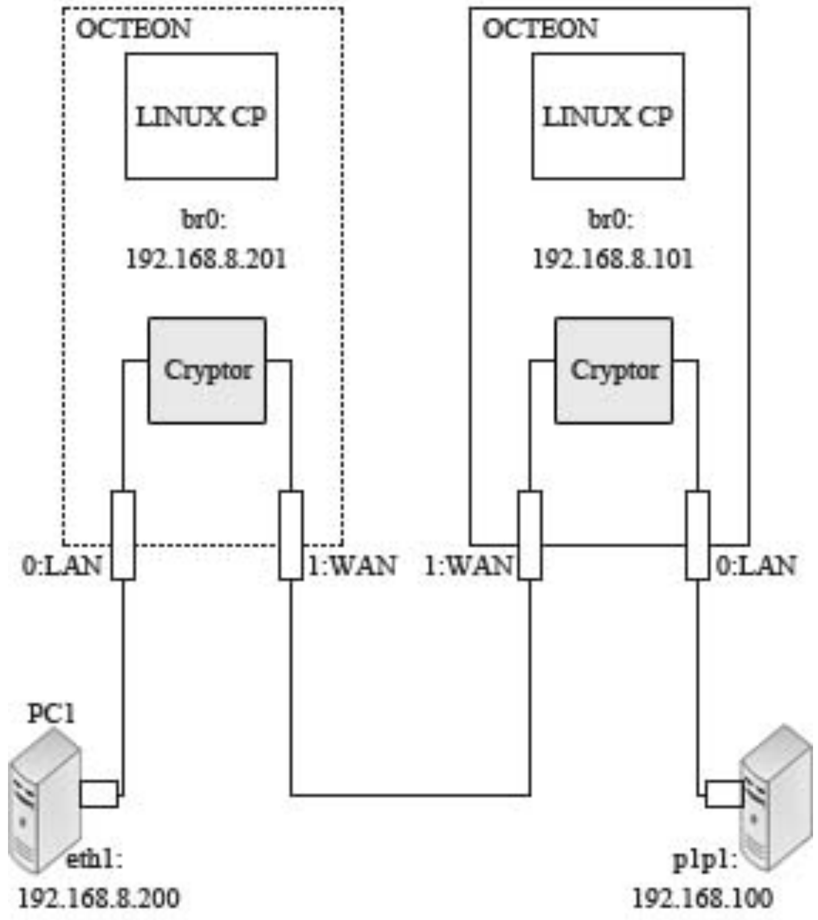

**Fig. (3).** Topology fortest.

The flow control module: the flow control module is put between the entering of the data package and SPDB. Because the data package belonging to the same data flow has the same five elements (source/destination port, source/ destination address, protocol), these data packages apply to the same SP. So we needn't search for all the data packages in SPDB. Only after the establishment of the data flow, to retrieve once is enough. If the specified motion is block, the data package will be abandoned; if the specified motion is pass, the data package will be transmitted to the output management module; if the specified motion is encryption, all the data packages belonging to the same flow, as well as the encrypted secret key, will be transmitted to the encryption mode.

SADB/SPDB management module: we apply to assign a special memory from the global memory to store SADB and SPDB, that's sharing memory. SADB and SPDB can be visited directly by the data platform and the control platform. The data platform mainly serves as a search database for the flow control module, while the control platform mainly serves as an available API base for application programs. This API base provides application programs with functions such adding, deleting, updating as well as obtaining SADB and SPDB.

The structure design of SADB and SPDB is as follows:

```
typedef struct
{
policy_t security_policy_table [2] [2048]; //SPDB
tcc_key_t security_key_table [4096]; //SADB
} cx7211_csp_t;
```

The encryption module: the data package entering the encryption module will be encrypted according the global configuration. In the following steps, the encryption module will assemble ESP protocols and encapsulate data packages.

The encryption module: the encrypted data package received from the decryption port will be decrypted by the encryption module according to the global configuration. In the next step, this package will be assembled into the original protocol type and transmitted to the output management module.

If decryption or verification fails, the statistic counter will increase successively according to the log. The record of the log message will be confined to at most once per 10 ms.

**Table 1. Performance comparison.**

| Input Size (bytes) | Octeon Cycles of AES-GCM Encrypt/decrypt |
|---|---|
| 32 | 358 |
| 256 | 1677 |
| 512 | 3141 |
| 1024 | 6169 |
| 8192 | 51677 |

**Table 2. Theoretical maximum throughput.**

| Packet Size (bytes) | Input/Output Cycles | Flow Manager Cycles | Policy Searching Cycles | Encryption and Decryption | Compression and Decompression | Through -put |
|---|---|---|---|---|---|---|
| 66 | 500 | 500 | 2666 | 358 | 500 | 206 |
| 290 | 500 | 500 | 2666 | 1677 | 500 | 558 |
| 546 | 500 | 500 | 2666 | 3141 | 500 | 811 |
| 1058 | 500 | 500 | 2666 | 6169 | 500 | 1088 |
| 8226 | 500 | 500 | 2666 | 51677 | 500 | 1536 |

The output management module and the fragmentor module: the output management module puts the data package in the output-queueing *via* the PKO hardware in the physical port, and then sends it to the output port assigned by the data package. If the length of the data package is greater than the value of the configurated MTU, the fragmentor module will decompose the data package into several small data packages and send them out in turn.

## 5. PERFORMANCE ANALYSIS

Make performance analysis on the software in this software framework *via* constructing the testing environment. The experiment topology is as follows:

The most two time-consuming operations in the control platform are: searching SADB and SPDB as well as the encryption and decryption of the data package. After our testing, the time used to search one record in SADB and SPDB is about 40 Cycles. Suppose in the worst case, we have 2048 records to retrieve, which is from the No. 0 position in the array to the No. 2047 position.

We have to compare at least 1,000 times before a record is retrieved. Therefore, the minimum CPU time for each flow to retrieval is 40,000 Cycles. If there are 15 data packages in each flow, every data package will consume 2,666 Cycles. As for the encryption and decryption of the data package, the following Fig. (**3**) gives the time spent in encrypting and decrypting the data packages with different lengths.

In addition to the above two main operations, we have calculated by testing that the average time spent in processing each data package by the other operations in the platform is 500 Cycles. These operations include the composition and separation of the data package in Input/Output Management Module (Tables **1** and **2**), Flow Control Module, encryption/Decryption Module. Here, we assume there is no need to fragment and reassemble the data package. In the following figure, we have calculated the time spent in processing plaintext packages with different lengths. This figure also shows we can theoretically get the maximum throughput from the core with a frequency of 1.3 GHZ.

Here, the calculation of the maximum throughput is based on the following rule:

packets_per_second = OCTEON_clock_rate / cycles_per_packet;

max_throughput = (packet_size + preamble + FCS + IPG) * 8 * packets_per_second;

The preamble value of the data package is 8 bytes, FCS is 4 bytes and IPG is 12 bytes. In the end, we can get the actual maximum throughput as Table **3**.

## CONCLUSION

In this paper, we have provided a highly-efficient software framework of IPSec crypto offload, and discussed in detail the design of this software framework, and presented the realization of the data platform and the control platform.

**Table 3. Actual maximum throughput.**

| Packet Size (bytes) | Input/Output Cycles | Flow Manager Cycles | Policy Searching Cycles | Total Cycles | Maximum Throughput Per Octeon Core (Mbps) |
|---|---|---|---|---|---|
| 66 | 500 | 500 | 2666 | 3666 | 255 |
| 290 | 500 | 500 | 2666 | 3666 | 890 |
| 546 | 500 | 500 | 2666 | 3666 | 1617 |
| 1058 | 500 | 500 | 2666 | 3666 | 3069 |
| 8226 | 500 | 500 | 2666 | 3666 | 23404 |

In addition, we have also made performance testing and analysis based on the advanced OCTEON chip platform in this field. But due to the limited space, many details haven't been discussed, such as the design of SADB and SPDB, the state of data structure, the general situation and use of the acceleration and encryption hardware provided by OCTEON processing, and so on. At the end of this paper, we have made a detailed performance analysis on the design of this software framework, with the result that this software has excellent performance. According to the upgrade of the OCTEON chip, how to make our software support more different encryption/decryption algorithms will be our future research direction.

## CONFLICT OF INTEREST

The author confirms that this article content has no conflict of interest.

## ACKNOWLEDGEMENTS

This paper belongs to the project of the Chongqing Normal University foundation project (13XWB008) and the Science and Technology Project Affiliated to the Education Department of Chongqing Municipality (KJ1400508).

## REFERENCES

[1] N. Doraswamy, and D. Harkins, "*IPSec: new generation of Internet security standards*", Beijing: China Machine Press, 2000

[2] National Institute of Standards and Technology (NIST). *Advanced encryption standard (AES).* FIPS Publication 197, Nov. 2001.

[3] W. Zhang, "Application and network scheme of IPSec VPN", *Information and Communications*, vol. 4, pp. 129-130, 2011.

[4] J. Xu, and S. Chen, "Comparison and analysis of IPSec-based and SSL-based VPN", *Comput. Eng. Des.*, vol. 25, no. 4, pp. 586-588, 2004.

[5] J. Lan, and C. Lin, "Analysis and Suggestion on the AH and ESP in IPSec", *Compuuter Technology and Development*, vol. 20, no. 10, pp. 167-170, 2010.

[6] G. Xiao, "Research and simulation of Qos of IPSec virtual private network", *Computer Simulation*, vol. 26, no. 8, pp. 137-142, 2009.

[7] H. Chen, Q. Zhou, and Y. Zhang, "Application and Implementation of VPN based on IPSec", *Manufactureing Automation*, vol. 33, no. 2, pp. 70-72, 2011.

[8] D. Box, D. Ehnebuske, and G. Kakivaya, "Simple object access protocol (SOAP) 1.1 [EB/OL] May 2000. Available at: http://www.w3.org /TR/ SOAP"

[9] H. Chen, Q. Zhou, and Y. Zhang, "Application and implementation of VPN based on IPSec", *Manufacturing Automation*, vol. 33, no. 2, pp. 70-72, 2011.

[10] T. Aura, "Cryptographicaly generated address (CGA). RFC3972", *Internet Engineering Task Force*, 2005.

Received: November 26, 2014 Revised: January 07, 2015 Accepted: January 20, 2015