Simplified IPSec Protocol Stack for Micro Server

Nguyen Thanh Hoa[1], Kensuke Naoe[1], and Yoshiyasu Takefuji[2]
(Corresponding author: Nguyen Thanh Hoa)

Graduate School of Media and Governance, Keio University, Endo 5322, Fujisawa, Kanagawa, Japan[1]
Faculty of Environment and Information Studies, Keio University, Endo 5322, Fujisawa, Kanagawa, Japan[2]
(Email: {hoant, naoe, takefuji}@sfc.keio.ac.jp)

(Received Oct. 26, 2008; revised and accepted June 27, 2009)

Abstract

In this paper, we propose a simple IPSec protocol stack for Micro Server. We proposed an implementation of IPSec protocol stack which is constructed by Encapsulating Security Payload (ESP) protocol with Advanced Encryption Security (AES) encryption scheme, whereas authentication using MD5 algorithm is optional. Researchers have focused on creating a small system composed of sensors and a Micro Server where it has a small sized memory, multi-function, low cost, but without security consideration. The security problem in the Micro Server is a challenging task because of the very limited flash memory.Here, we have implemented the AES function as 2.704 Kbytes and the ESP protocol with this encryption function as 3.822Kbytes of code. Therefore, the proposed method has less than 4Kbytes in code size. Even including the authentication using MD5, the file size is less than 7Kbytes although this is still optional. In our proposed method, we have focused on implementing the encapsulation of the payload and ignored the key exchange procedure to simplify the secure communication compared to conventional IPSec protocol stack.

Keywords: 8-bit micro-controller, IPSec, micro server, security, sensor

1 Introduction

In the world, many people gather their strength and intelligence to have fast and accurate information. Therefore, information becomes the target that everyone wants to pursue. On the other hand, everyone wants to keep their own information secret. Nowadays, with the development of technology and science, we can make small sensors and Micro Servers very easily. These small and cheap sensors and Micro Server are deployed in many useful, low cost applications. Security is an important issue when these devices are used in health care applications, home appliances and many others. However, researchers design sensors and Micro Server with purpose of small size, low cost rather than security. Security is challenging tasks because of very small processors and very limited memory [13].

Cryptography is the art of secret writing. Cryptography guarantee security properties such as authentication or secrecy in the information exchange between users and server. This paper analyzes normal security methods using cryptography and then proposes a simple IPSec protocol that can protect very small sensors and Micro Server. In the implementation, we have established ESP protocol with code size less than 7Kbytes. This simple IPSec protocol is very useful for security connection between small devices and Internet. IPv6 is the "next generation" protocol to replace the current version Internet Protocol IPv4 [12]. In IPv6, IP security protocol (IPSec) is a mandatory feature. RFC 2460 [9] states that a "full implementation of IPv6" includes implementation of the authorization header (AH) and encapsulating security payload (ESP) [14]. The simple IPSec protocol helps very small devices to connect safely to Internet IPv4 and is good referred materials to improve and apply for IPv6.

2 Background

2.1 The Micro Server

We made a Micro Server gadget that followed the design of Takefuji [19] that is shown in Figure 1. This Micro Server uses 8bit micro-controller Atmega168 with 16 Kbytes flash memory. Adam Dunkel, the author of the simple TCP/IP stack embedded inside this Micro Server with size of flash memory is about 8466 bytes [20].

The limitation of the Micro Server is that it has very small memory (only left 7 Kbytes for security function) and limited processing: 512 Bytes EEPROM Data Memory, 1024 Bytes SRAM Data Memory, 32 MCU General Purpose Registers (Accumulators), 0 - 20 MHz Specific Clock Frequency Supply, 2.7 - 5.5 v Voltage. We will use this Micro Server to test the implementation of a simple IPSec protocol that we will describe in the following sections.

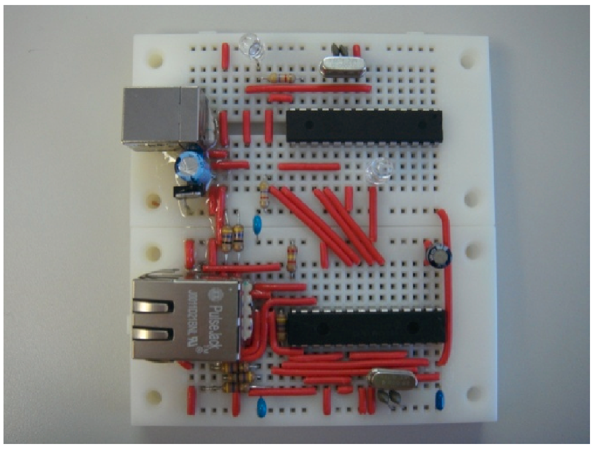

Figure 1: Micro server

2.2 The Security Methods

To transfer the information from sensor and Micro Server to PC, we can establish a simple TCP/IP stack on these devices. We have to use the security methods to prevent hackers who want to get the information or attack Micro Server. Cryptographic protocols are good security methods with high confidence. There are two main types of security methods for TCP/IP stack: security for application layer and security for network layer. However, Micro Server using 8-bit micro-controller has limited flash memory, small processor so we analyze theses properties to choose a suitable method for security.

2.2.1 Security for Application Layer

To secure an application layer, we can establish SSH - Secure Shell protocol by using 3DES for encryption and RSA for authentication. However, implementing public key cryptography for Micro Server is difficult if we have only 7 Kbytes flash memory.

Public key cryptography has high security and convenience. It provides digital signatures for encryption and authentication. We can apply these public key cryptographic algorithms to secure application layer. Public key cryptographic algorithms have public keys and private keys. Other sensors or Micro Servers encrypt data by using public keys and send to receiver. Only the device with the proper private key can decrypt data correctly.

RSA is well-known algorithm and is used in many of the public key systems. However, the significant parameters such as the speed of execution, the difficulty of key generation, establishment of system parameters and the size of data to be stored in the memory is too large and makes RSA not suitable for securing these small sensors and Micro Server [2].

2.2.2 Security for Network Layer

IPSec is a suitable protocol for securing network connections but it is complex protocol. This provides the ability to encrypt any higher layer protocol and authenticating each IP packet. IPSec offers the greatest flexibility of all the existing TCP/IP cryptosystems. We can see the complexity of IPSec protocol in the Figure 2 [3]: processing key exchange; processing Security Policy, Security Association (by SPD, SAD) and two protocols AH - Authentication Header, ESP - Encapsulating Security Payload.

IPSec defines SA - Security Association. SA is a relationship between two or more entities that present how the entities use IPSec to communicate securely. When IPSec is required, the end points have to determine security parameters such as which algorithms to use (for example, DES or AES for encryption, MD5 or SHA for integrity). SA is defined by the packet's destination IP address and a SPI - Security Parameter Index. There are two modes of IPSec: transport mode and tunnel mode. In transport mode, IPSec data field begins with higher level packet headers (ICMP, TCP or UDP). Tunnel mode is similar with traditional VPN; IPSec data field begins with an entirely new IP packet header.

In IPSec, there are two main protocols to provide packet-level security: AH - Authentication Header and ESP - Encapsulating Security Payload. AH protocol provides integrity, authentication and non repudiation. The AH can protect replay attacks by using sliding window technique and discarding old packets. In AH transport mode, IP packet include the new AH header and full IP header that is shown in Figure 3.

ESP protocol provides confidential protection, authentication and integrity. This protocol has encryption and authentication functions. Authentication is optional but if we use encryption without authentication then ESP protocol is insecure and crackers can attack this connection. IPSec transport mode is shown in Figure 4.

Although a normal IPSec protocol is more complex than SSH, but is more feasible because we propose a simple IPSec protocol in the following Section 3.

3 The Proposed Simple IPSec

Micro-controller is used for creating sensors or Micro Server and has a limited memory and small processors, for example 8bit-micro-controller has only 16 Kbytes flash memory. Therefore, we have very limited memory for establishing IPSec protocol. If we have only 7 Kbytes flash memory, then we need a very simple IPSec protocol to secure connections between client and Micro Server or connections from sensors to PCs.

We have a simple TCP/IP stack extending figure that includes IPSec protocol as Figure 5 [20].

Although IPSec protocol is very complex: processing key exchange; processing Security Policy, Security Association and two protocols AH, ESP as we mentioned above, we can simplify this protocol by reducing functions and

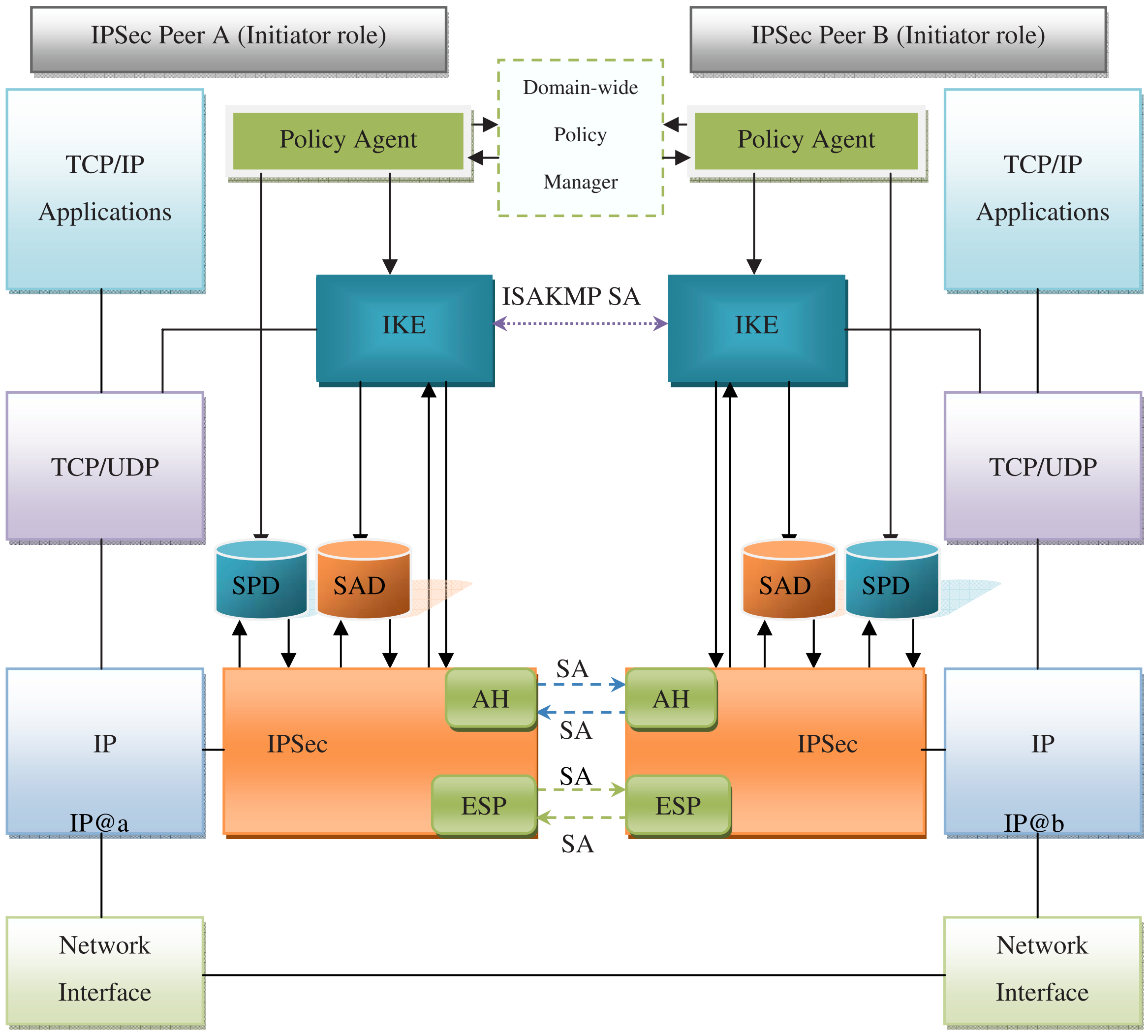

Figure 2: The IPSec architecture

optimize coding to have smallest code size. There has been significant debate about the necessity for AH, which provides only integrity protection, since ESP can provide integrity protection or encryption or both.

The integrity protection provided by AH extends to portions of the IP header, whereas ESP's integrity protection is only of the payload. The opponents of AH argue that it is unnecessary to protect the IP header, and if it were necessary, could be provided by tunnel mode. The debate has been heated and the issues are complex [6, 16].

In this case, ESP protocol is enough for securing connections between peers so establishing a simple IPSec protocol is only the simple implementation ESP protocol. By this way, we can establish a secure VPN - virtual private network that require both authentication and encryption. To save memory, we can implement key and SPI, IV parameters in the program and do not need processing key exchange, processing SA in Micro Server. In the future, if we have enough memory space, we will process these functions.

RFC 4303 standard [18] describes ESP in detail. ESP has both encryption and authentication, therefore ESP is complex protocol. We can establish ESP protocol with one encryption algorithm which has high confidence and a hash function for authentication which has smallest code size.

DES is well-known in symmetric key cryptography and is a default algorithm for normal IPSec protocol but DES now is insecure as mentioned in many studies [8, 15]. Unlike DES, AES algorithm is fast in both on software and hardware, easy to implement and requires little memory. In recent years, AES is deployed on a large scale [1]. Therefore, we choose only AES algorithm for encryption in ESP protocol.

Figure 6 shows the simple IPSec protocol using simple ESP which uses AES for encryption and hash function MD5 for authentication.

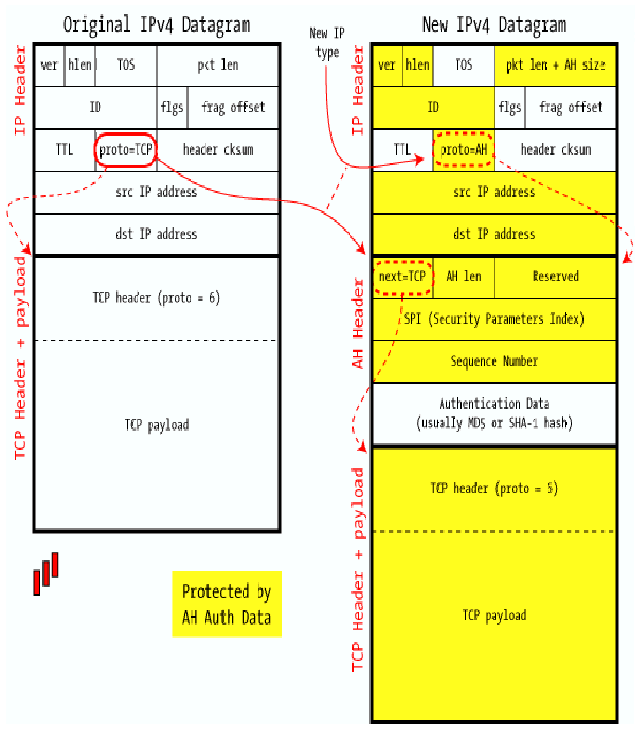

Figure 3: IPSec in AH transport mode (Source: An illustrated guide to IPSec)

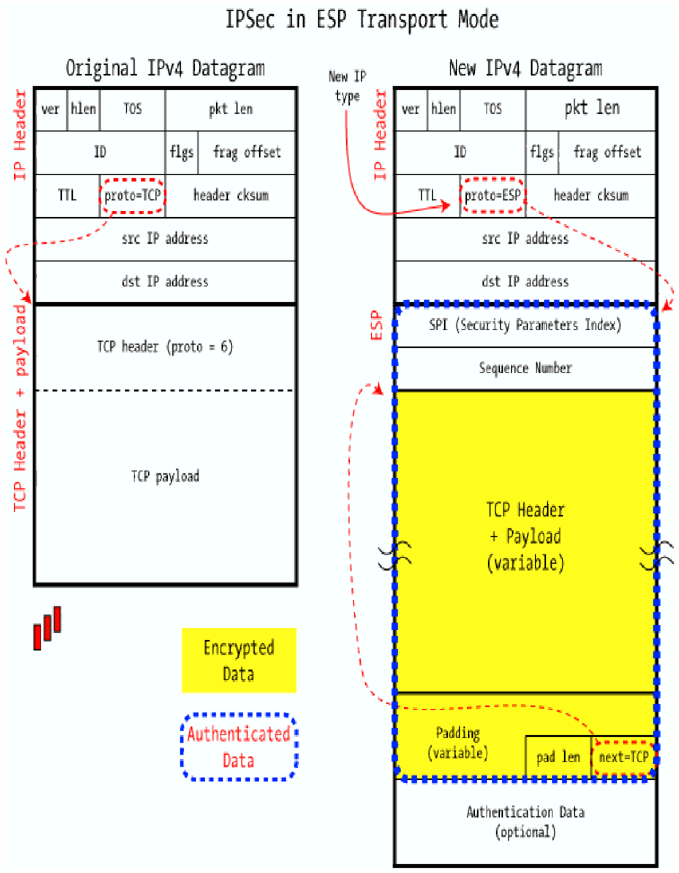

Figure 4: IPSec in ESP transport mode (Source: An illustrated guide to IPSec)

4 Implementation & Experiment Results

There are 4 steps to establish this simple IPSec protocol for Micro Server:

1) Implementing AES algorithm.
2) Implementing MD5 hash function.
3) Establishing ESP protocol: processing ESP header, IP header and encryption/decryption higher payload, implementing authentication.
4) Testing the securable connection between client and Micro Server when a simple IPSec is established in Micro Server and client use a normal IPSec protocol.

We have implemented Steps 1, 2 and 3: implemented AES algorithm in 2704 bytes (2.704 Kbytes) and MD5 hash function in 2144 bytes (2.144 Kbytes). Also we have established ESP protocol (only encryption) with code size of 3822 bytes (3.822 Kbytes). Now we are testing this simple IPSec (ESP encryption) with a normal IPSec protocol in the client side. The final size of the simple IPSec program is less than 7 Kbytes. In the client's side, we use Openswan to create a full IPSec protocol.

4.1 AES Implementation

We followed the standard of Advanced Encryption Standard - AES to implement this algorithm, Federal Information Processing Standards Publication (FIPS PUB) is issued by the National Institute of Standards and Technology (NIST) [5].

The main AES Encryption is shown as following code:

```
void encryptAES(u8 *data, u8 *key, u32 data_length, u8
*cipher)
   {
    u32 i;
    u8 *s;
    u8 rnd_key[16];
    u8 rnd_con[4] = 0x01, 0x00, 0x00, 0x00;
    u8 j;
    for(i=0 ; i ¡ data_length ; i+=16)
     {
      s = data+i;
      rnd_con[0] = 0x01;
      for(j=0 ; j¡16 ; j++)
      rnd_key[j]=key[j];
      // K = 128, Nr = 10, Nk = 4.
       for(j = 0; j ¡ 9; j++)
        {
         addRoundKey(s, rnd_key);
```

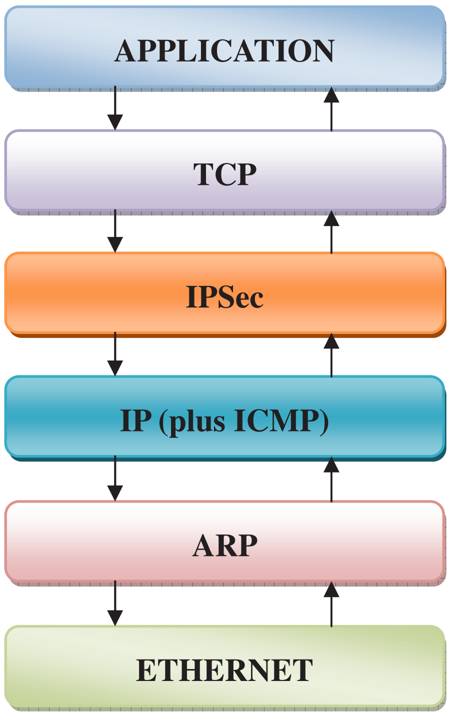

Figure 5: A simple TCP/IP stack including IPSec

```
        substituteByte(s);
        shiftRow(s);
        mixColumn(s);
        keyExpansion(rnd_key, rnd_con);
      }
    addRoundKey(s, rnd_key);
    substituteByte(s);
    shiftRow(s);
    keyExpansion(rnd_key, rnd_con);
    addRoundKey(s, rnd_key);
    for(j=0 ; j¡16 ; j++)
      cipher[j] = s[j];}
}
```

The AES decryption function is:
*void decryptAES(u8 *data, u8 *key, u32 data_length, u8 *decipher)*

```
  {
    u32 i;
    u8 *s;
    u8 j;
    u8 rnd_key[16];
    u8 rnd_con[4] = 0x01, 0x00, 0x00, 0x00;
    for(i=0 ; i ¡ data_length ; i+=16)
      {
        s = data+i;
        rnd_con[0] = 0x01;
        calLastRoundKey(rnd_key, key, rnd_con);
        for(j=0 ; j¡16 ; j++)
        s[j] = s[j]r̂nd_key[j];
```

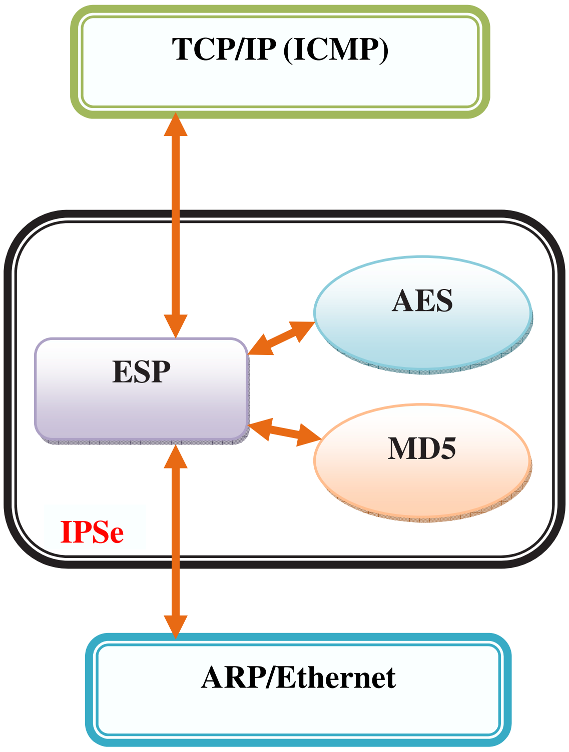

Figure 6: A simple IPSec protocol

```
        // K = 128, Nr = 10, Nk = 4.
        for(j = 9; j ¿ 0; j–)
          {
            invShiftRow(s);
            invKeyExpansion(rnd_key,rnd_con);
            invSubstituteByte(s);
            invAddRoundKey(s, rnd_key);
            invMixColumn(s);
          }
        invShiftRow(s);
        invSubstituteByte(s);
        invAddRoundKey(s, rnd_key);
        for (j = 0; j ¡ 16; j++)
          decipher[j] = s[j];
      }
  }
```

We tested this algorithm implementation by turn on or turn off of LED 1, 2 in the Micro Server gadget.

```
// AES test:
    u8 key[16] = { 0x00, 0x01, 0x02, 0x03, 0x04, 0x05,
0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e,
0x0f};
    u8 data[16] = { 0x00, 0x11, 0x22, 0x33, 0x44, 0x55,
0x66, 0x77, 0x88, 0x99, 0xaa, 0xbb, 0xcc, 0xdd, 0xee,
0xff};
    u8 store[16] = { 0x00, 0x11, 0x22, 0x33, 0x44, 0x55,
0x66, 0x77, 0x88, 0x99, 0xaa, 0xbb, 0xcc, 0xdd, 0xee,
```

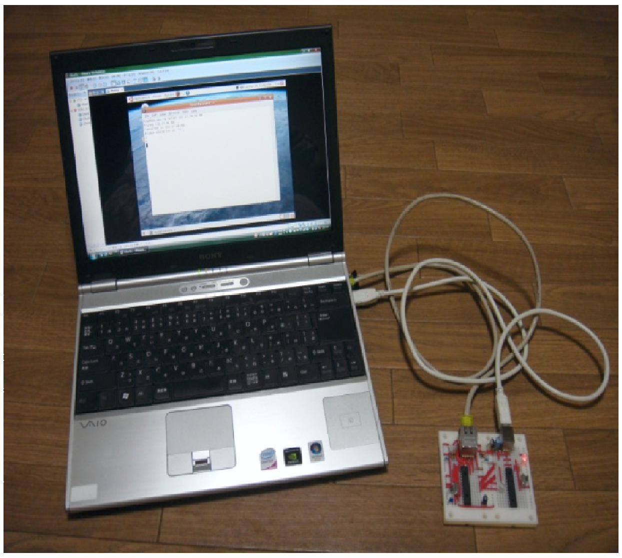

Figure 7: AES testing environment for Micro Server

```
0xff};
    u8 cipher[16] = { 0x00, 0x01, 0x02, 0x03, 0x04, 0x05,
0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e,
0x0f};
    u8 decipher[16] = { 0x01, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d,
0x00, 0x00};
    int k;
    encryptAES(data, key, 16, cipher);
    decryptAES(cipher, key, 16, decipher);
    int tmp = 2;
    for (k = 0; k ¡ 16; k++){
      if (tmp == 1)
        { // turn on LED 1
          DDRD= (DDRD — 0x80);
          PORTD= (PORTD — 0x80); }
      else if (tmp == 2)
        { // turn on LED 2
          DDRC = (DDRC — 0x01);
          PORTC = (PORTC — 0x01); }
      if (decipher[i] == store[i])
        tmp = 1;
      else
        tmp = 0;
    }
```

We had experiment result of this implementation: LED 1's light is turned on as Figure 7.

4.2 IPSec Protocol Implementation

In Micro Server, we have established IPSec protocol that has only ESP protocol with AES in CBC mode [17] for encryption. We have diagrams of simple IPSec processing as the following Figure 8 and Figure 9.

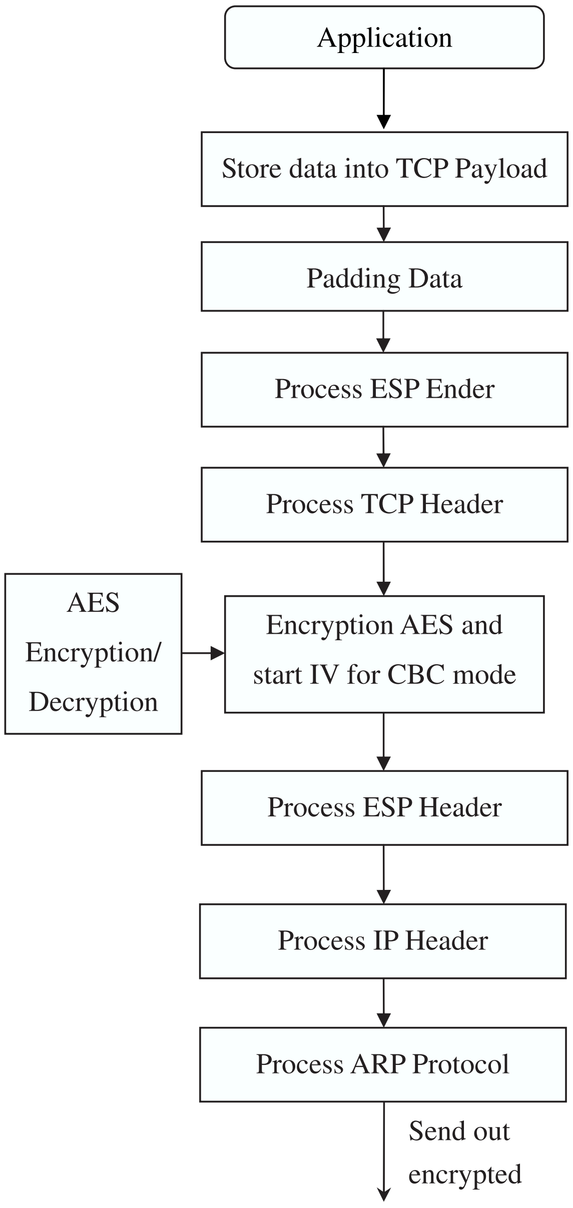

Figure 8: Encryption process in the simple IPSec protocol

To simplify IPSec protocol, we fixed the value of SPI and IV - Initialization Vector. We saved memory by not creating SAD and SPD. This IV is used for AES CBC mode in ESP protocol.

```
    u8_t iv_tmp[16] = {0xD4, 0xDB, 0xAB, 0x9A, 0x9A,
0xDB, 0xD1, 0x94, 0xD3, 0xDA, 0xAA, 0x99, 0x99,
0xDA, 0xD0, 0x93};
    BUF → spi = 0x00001018;
```

We also did not create IKE protocol because we want to skip this protocol to save memory for Micro Server. In the decryption process of IPSec, we have:

```
enc_ptr += tmpNo;
tcp_ptr += tmpNo;
while (tmpNo ≥ 0)
```

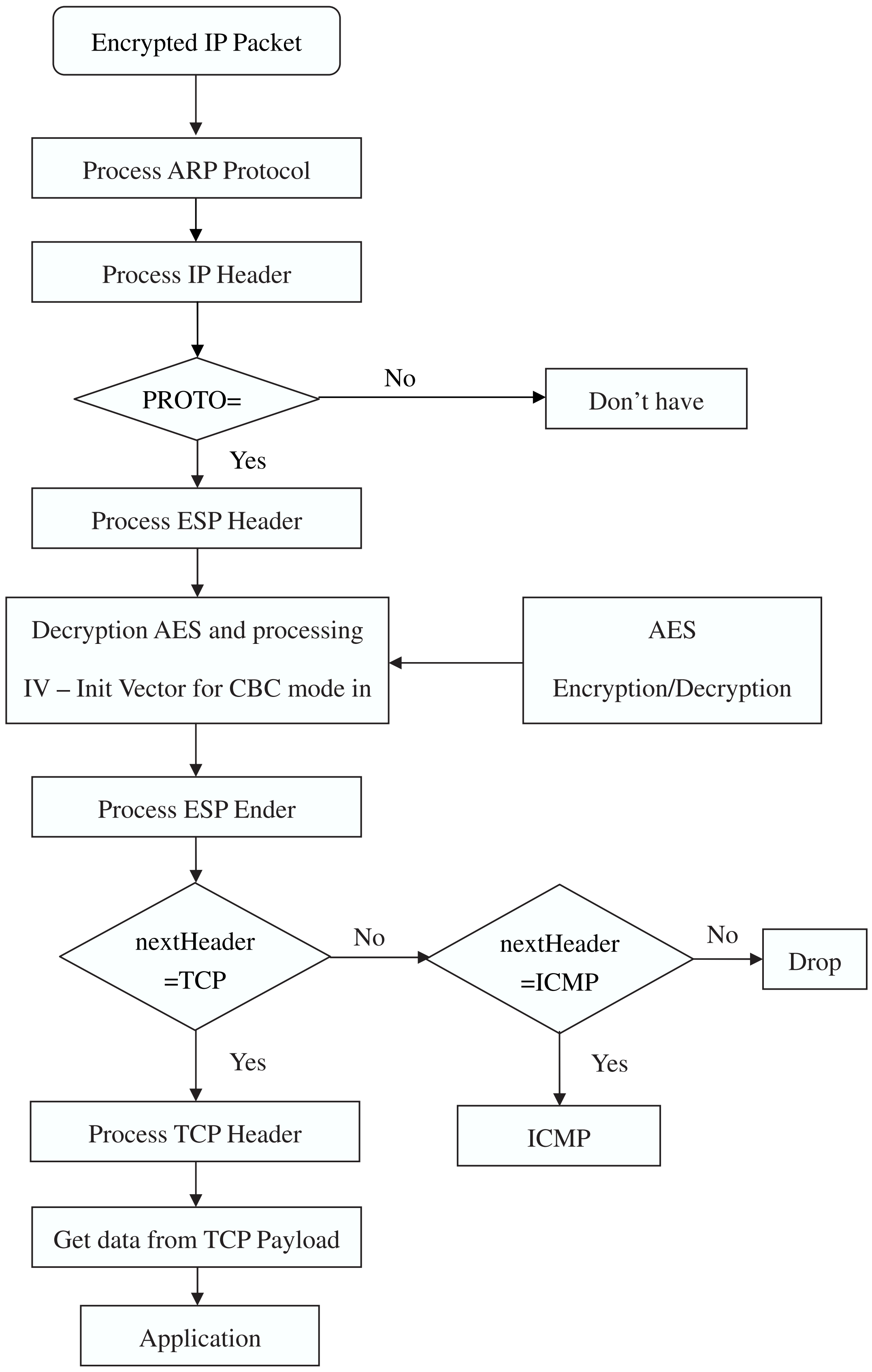

Figure 9: Decryption process in the simple IPSec protocol

```
  {
   decryptAES(enc_ptr, keyEnc, 16, tcp_ptr);
   if(tmpNo == 0)
     iv_pro = (const u32*) iv_sto;
    else
     iv_pro = (const u32*) (enc_ptr - 16);
    // CBC Processing
    *((u32 *)(&tcp_ptr[ 0])) ^= iv_pro[0];
    *((u32 *)(&tcp_ptr[ 4])) ^= iv_pro[1];
    *((u32 *)(&tcp_ptr[ 8])) ^= iv_pro[2];
    *((u32 *)(&tcp_ptr[12])) ^= iv_pro[3];
    enc_ptr -= 16;
    tcp_ptr -= 16;
    tmpNo -= 16;
  }
```

In encryption process of IPSec, we have:

```
u8_t iv_tmp[16] = {0xD4, 0xDB, 0xAB, 0x9A, 0x9A,
0xDB, 0xD1, 0x94, 0xD3, 0xDA, 0xAA, 0x99, 0x99,
0xDA, 0xD0, 0x93};

iv_sto = &iv_tmp[0];
while (tmpNo ¡ uip_len)
  {
   if(tmpNo == 0)
    iv_pro = (const u32*) iv_sto;
   else
    iv_pro=(const u32*) (enc_ptr-16);
   *((u32 *)(&enc_ptr[ 0])) = iv_pro[0]^*((const u32
   *)(&tcp_ptr[0]));
   *((u32 *)(&enc_ptr[ 4])) = iv_pro[1]^*((const u32
   *)(&tcp_ptr[4]));
   *((u32 *)(&enc_ptr[ 8])) = iv_pro[2]^*((const u32
    *)(&tcp_ptr[8]));
   *((u32 *)(&enc_ptr[12])) = iv_pro[3]^*((const u32
    *)(&tcp_ptr[12]));
   encryptAES(tcp_ptr, keyEnc, 16, enc_ptr);
   tcp_ptr+=16
   enc_ptr+=16;
   tmpNo+=16;
  }
```

In the client side, we will establish IPSec protocol by using Openswan installed in Ubuntu on VMware. We changed ipsec.conf of Openswan manually [10].

```
# /etc/ipsec.conf - Openswan IPsec configuration file
# RCSID $Id: ipsec.conf.in,v 1.15.2.6 2006-10-19
03:49:46 paul Exp $
# Manual: ipsec.conf.5
version 2.0
# conforms to second version of ipsec.conf specification
# basic configuration
config setup
     klipsdebug = none
     plutodebug = none
     uniqueids = yes
# Add connections here
# sample VPN connections, see /etc/ipsec.d/examples/
conn%default
     keyingtries = 0
# transport ESP with manual setting - AES for encryp-
tion.
# between uip in Micro Server (133.27.66.66) & linux
machine "hoa" (133.27.66.1) conn secuip
conn secuip
     authby = manual
     left = 133.27.66.66
     leftid = uip
     right = 133.27.66.1
     rightid = @hoa
     esp = aes128
     spi = 0x1018
     espenckey =
0x00010203_04050607_08090a0b_0c0d0e0f
     auto = add
# Disable Opportunistic Encryption include
/etc/ipsec.d/examples/no_oe.conf
```

We have to use software (Openswan) to create IPSec protocol in the client side and test with Micro Server. Normally, this software doesn't allow skipping IKE protocol. Therefore, there are two ways to solve this problem: changing type of micro-controller in Micro Server or finding software that can skip IKE protocol. For the former, we will create small IKE protocol in Micro Server. The current Micro Server has limited micro-controller with 16 Kbytes and it is not enough. We need to change type of this micro-controller to have memory more than 25 Kbytes. For the latter, we will find how to ignore IKE protocol in the client side and test with simple IPSec protocol in Micro Server.

5 Conclusion and Future Work

In this paper, we have analyzed security methods for Micro Server which has limited memory and small processor. By this analysis, we can see that IPSec is good choice for Micro Server security. We proposed a very simple IPSec protocol for establishing a secure layer in a simple TCP/IP stack. A full IPSec protocol has many functions and is complex. For us, the simple IPSec protocol need only ESP protocol with only AES algorithm for encryption and only MD5 algorithm for authentication. This simple IPSec protocol can establish secure connections which satisfy security requirement of Micro Server or sensors. We have implemented AES algorithm in 2704 bytes, MD5 in 2144 bytes and simple ESP protocol (with encryption). We have established a simple IPSec protocol that has only AES encryption with code size of 3822 bytes. The final size of the simplified IPSec program is less than 7 Kbytes.

In the future, we will find the suitable method to test and debug the secure connection between this simple IPSec in Micro Server with a normal IPSec protocol of client to realize a very small security system for Micro Server.

References

[1] *Advanced Encryption Standard.* (http://en.wikipedia .org/wiki/Advanced_Encryption_Standard)

[2] F. Amin, A. H. Jahangir, and H. Rasifard, "Analysis of public-key cryptography for wireless sensor networks security," pp. 530-535, 31 July 2008.

[3] Atmel Corporation, *Atmega168 Datasheet.* (http://www.datasheetcatalog.org/datasheet/atmel /2545S.pdf)

[4] A. Dunkels, and J. Alonso, *Making TCP/IP Viable for Wireless Sensor Networks*, Thiemo Voigt Swedish Institute of Computer Science.

[5] Federal Information Processing Standards Publication 197, *Announcing the Advanced Encryption Standard (AES)*, Nov. 26

[6] N. Ferguson, and B. Schneier, *A Cryptographic Evaluation of IPSec*, Apr. 1999. (http://www.counter -pane.com/ipsec.html)

[7] S. Friedl, *An Illustrated Guide to IPSec.* (http:// unixwiz.net/techtips/iguide-ipsec.html)

[8] M. Hellman, "DES will be totally insecure within ten years," *IEEE Spectrum*, vol. 16, no. 7, pp. 31-41, July 1979.

[9] *Internet Protocol, Version 6 (IPV6) Specification*, RFC 2460, 1998. (http://www.faqs.org/rfcs/rfc2460 .html)

[10] *IPSec.conf - IPSec Configuration and Connections Reference.* (http://man.free4web.biz/man5/ipsec. conf.5.html)

[11] *IPSec Guide Architecture & Traffic Processing.* (http://www.tech-invite.com/Security/Ti-IPSec-archi.pdf)

[12] *IPv6 Information Page.* (http://www.ipv6.org/)

[13] K. Lorincz, and D. J. Malan, T. R. F. F. Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, and S. Moulton, "Sensor networks for emergency response: Challenges and opportunities, *IEEE Pervasive Computing*, vol. 3, no. 4, pp. 16-23, 2004.

[14] C. Metz, "Guest editor's introduction: Moving toward an IPV6 future," *IEEE Internet Computing*, vol. 7, no. 3, pp. 25-26, May/June 2003.

[15] A. E. Pascual, *History of DES.* (http://www.it46.se /downloads/courses/security/en/02_Cryptography/ en_security_B2B_historyDES_slides_escuderoa.pdf)

[16] R. Perlman, and C. Kaufman, "Key exchange in IPSec - Analysis of IKE," *IEEE Internet Computing*, vol. 4, no. 6, pp. 50-56, 2005.

[17] *The AES-CBC Cipher Algorithm and Its Use with IPSec*, RFC 3602, 2003. (http://www.faqs.org/rfcs/ rfc3602.html)

[18] *IP Encapsulating Security Payload (ESP)*, RFC 4303, 2005. (http://www.ietf.org/rfc/rfc3686.txt)

[19] Y. Takefuji, Takefuji Lab. (http://www.neuro.sfc .keio.ac.jp/)

[20] *The open-source uIP TCP/IP stack.* (http://www. dunkels.com/adam/uip)

[21] *Openswan.* (http://www.openswan.org/)

Nguyen Thanh Hoa is a second year Master student of Graduate School of Media and Governance at Keio University in Japan. From 1999 to 2002, she learned in high school for talented Mathematics students of National University in Vietnam. She received the bachelor degree from Posts and Telecommunications Institute of Technology (PTIT) in 2007. She got the second prize for student's research in PTIT in 2005. From 2007, she has worked for Research Institute of Post and Telecommunication. Her major is cryptography, security protocols and internet gadgets. Her research interests are network security, network protocols, ubiquitous system, sensors network and transmit codes. Besides, she is interested in searching engineering, maps processing, mobile games, 3D environment, signaling processing for communications and telecommunication architecture.

Kensuke Naoe is a Ph.D. candidate, graduated from the Faculty of Environment and Information Studies of Keio University in 2002. He received the Master degree from the Graduate School of Media and Governance at Keio University in 2004. His major is artificial neural network and information security. His interested research areas are digital information hiding, machine learning, network intrusion detection and malware detection.

Yoshiyasu Takefuji is a tenured professor at the Faculty of Environment and Information Studies of Keio University, since April 1992 and was on tenured faculty of Electrical Engineering at Case Western Reserve University, since 1988. Before joining Case, he taught at the University of South Florida and the University of South Carolina. He received his BS (1978), MS (1980), and Ph.D. (1983) from Electrical Engineering from Keio University. His research interests focus on neural computing, security, internet gadgets, and nonlinear behaviors. He received the National Science Foundation/Research Initiation Award in 1989, the distinct service award from IEEE Trans. on Neural Networks in 1992, and has been an NSF advisory panelist. He has received the best paper award from AIRTC in 1998 and a special research award from the US air force office of scientific research in 2003. He is currently an associate editor of International Journal of Multimedia Tools and Applications, editor of International Journal on Computational Intelligence and Applications, and editor of International Journal of Knowledge-based intelligent engineering systems. He was an editor of the Journal of Neural Network Computing, an associate editor of IEEE Trans. on Neural Networks, Neural/parallel/scientific computations, and Neural computing. He has published more than 120 journal papers and more than 100 conference papers.