I. J. Computer Network and Information Security, 2013, 11, 36-41
Published Online September 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2013.11.05

Triple Layered Encryption Algorithm for IEEE 802.11 WLANs in E-Government Services

M A Kabi and K A Sayeed
Department of Electrical Engineering and Computer Science, North South University
Dhaka, Bangladesh

M A Matin
Institut Teknologi Brunei, Bandar Seri Begawan, Brunei Darussalam
si.ijcnis@gmail.com

T Mehenaz
Dept. of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET), Bangladesh

M Kamruzzaman
Dept. of Computer Science and Engineering
Chittagong University of Engineering and Technology (CUET), Bangladesh

Abstract — Wireless local area network (WLAN) can provide e-government services at all levels, from local to national as the WLAN enabled devices have the flexibility to move from one place to another within offices while maintaining connectivity with the network. However, government organizations are subject to strict security policies and other compliance requirements. Therefore, WLAN must ensure the safeguard the privacy of individual data with the strictest levels of security. The 802.11 MAC specifications describe an encryption protocol called Wired Equivalent Privacy (WEP) which is used to protect wireless communications from eavesdropping. It is also capable of preventing unauthorized access. However, the WEP protocol often fails to accomplish its security goal due to the weakness in RC4 and the way it is applied in WEP protocol. This paper focuses the improvement of existing WEP protocol using the varying secret key for each transmission. This will remove the insecurities that currently make the RC4 unattractive for secured networking and this will add further cryptographic strength if applied to Rijndael algorithm. Our result shows that the proposed algorithm is more suitable for small and medium packets and AES for large packets.

Index Terms — IEEE 802.11 WLAN; Network security; WEP protocol; RC4; symmetric encryption

I. Introduction

E-government facilitates citizens to receive public services. To provide such services, WLAN performs a vital role in streamlining operations inside government offices and institutions. It is expected that a better access to WLAN in the public sector can generate innovation chains, such as the health service. However, the citizens totally expect systems to be secured and privacy-preserving. Thus, security technologies, such as advanced encryption technique and access control must be integrated into e-government systems. The main security issue with wireless networks, especially radio networks, is that it intentionally radiates data over an area that may exceed the limits of the area the organization physically controls [1]. This characteristic of radio wave illustrates that wireless networks are subject to security risks [2]. In order to protect the data from eavesdroppers, various forms of encryption have been used. The IEEE 802.11 MAC specification describes a WEP protocol to make Wireless LAN communication as secure as wired LAN data transmissions. It uses a shared key mechanism with a symmetric cipher called RC4 [3]. However, there are several design flaws existing in WEP and it does not provide adequate protection. Therefore, Wi-Fi Protected Access (WPA) is a data encryption specification for 802.11 WLAN that replaces the weaker WEP which is further replaced with Wi-Fi Protected Access 2 (WPA2), an enhanced version of WPA. WPA2 is stronger than WPA because it uses Advanced Encryption Standard (AES) instead of RC4 or TKIP. Though WEP is the most vulnerable security measure in Wi-Fi protected mode, it dominated WLAN security for a quit long time. Moreover, many systems still support WEP for backward compatibility despite its known weakness and many people are still using it. The key in WEP that a client is using for authentication and encryption of the data stream must be the same key that the access point (AP) uses. This paper illustrates the idea that the key should not necessarily be the same always. By generating a Pointing Vector (PV) and a serial number of this PV for each transmission, a frame key can be made, which then will

be used for encrypting the message. The transmitted frame will include the encrypted data, the PV and the serial number (PVS). On the decryption side, PVs will be employed to generate the same frame key, and the PV will go through the key scheduling.

The rest of the paper is organized as follows. In section II, the proposed TLEP is explained. Section III holds frame key structure generation. Key generator is discussed in section IV. Then in Section V, key scheduling is explained. In section VI, simulation result is presented to show the performance comparison with the proposed approach. Finally, conclusion is drawn in Section VII.

II. Proposed Algorithm

The aim of WEP is to provide WLAN as secure as wired LAN. In WEP mechanism, a periodic IV (Initializing Vector) is generated [4]. Nevertheless, we call it by a different name in our algorithm, that is, the Pointing Vector (PV). The novelty does not lie in calling it by a different name; rather this vector with its three component bytes connects last two layers and determines the cipher. Our proposed TLEP can be explained following the Fig.1. Table 1 and Table 2 show different steps of encryption and decryption process of the proposed algorithm.

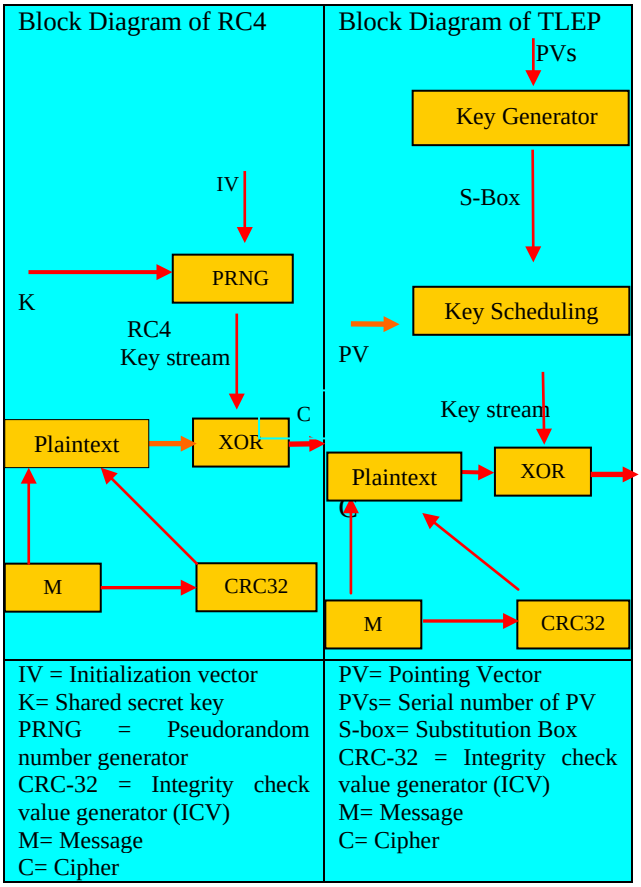

Figure 1. Block diagram of RC4 and our proposed algorithm.

TABLE 1: ENCRYPTION
Step 1: Create a message frame and calculate its checksum
Step 2: Generate a 24 bit long PV by a random number generating function and tag this PV with a serial number of 48 bits.
Step 3: Generate a frame key.
Step 4: Map the frame key in 255 byte long S box. This should be done by 42 complete cycles of the frame key plus the first three bytes.
Step 6: Generate a key sequence.
Step 7: Place the original message and its checksum together.
Step 8: EXOR this concatenation with the key sequence to create the cipher text.

TABLE 2: DECRYPTION
Step 1: Separate the PV from its serial number.
Step 2: Generate a frame key.
Step 3: Map the frame key in 255 byte long S box. This should be done by 42 complete cycles of the frame key plus the first three bytes.
Step 4: Generate a key sequence.
Step 5: EXOR the key sequence with cipher text to generate original message and its checksum.
Step 6: Separate the message from its checksum.
Step 7: Create checksum of the message.
Step 8: Check if this checksum matches with the earlier checksum.

III. Frame Key

The real catch in our algorithm is that the frame key will be generated from the PVs. Each instance of creating this PV generates 48 bits long serial number (PVs). The serial number of the PV will be input to the key generating function.

An integrity checksum value [ICV(M)] is calculated on the M using the cyclic redundancy check. The M concatenates ICV(P) to form the P. In other words, we have P = {M, ICV(M)}.

- A pointing vector (PV) is chosen as a 24 bit random number by the sending station. And a serial number of 48 bits (PVS) is also calculated for this IV.
- This serial number will be put into a key generator, making it possible to produce 248 (two hundred and eighty thousand billions) unique combinations. Each of these we will call a frame key. From this 48 bit long frame key we construct a 256 byte long S-box.
- The S-box and the PV is input to a key scheduling algorithm, the output of which is a keystream.
- The keystream and the P are Exored (Å) to obtain the cipher text (C). The real catch in our algorithm is that the frame key will be generated from the PVS. Each instance of creating this PV generates 48 bits long serial number (PVS). The serial number of the PV will be input to the key generating function.

An integrity checksum value [ICV(M)] is calculated on the M using the cyclic redundancy check. The M concatenates ICV(P) to form the P. In other words, we have P = {M, ICV(M)}.

- A pointing vector (PV) is chosen as a 24 bit random number by the sending station. And a serial number of 48 bits (PVS) is also calculated for this IV.
- This serial number will be put into a key generator, making it possible to produce 248 (two hundred and eighty thousand billions) unique combinations. Each of these we will call a frame key. From this 48 bit long frame key we construct a 256 byte long S-box.
- The S-box and the PV is input to a key scheduling algorithm, the output of which is a keystream.
- The keystream and the P are Exored (Å) to obtain the cipher text (C).

IV. Key Generator

In our algorithm, two frames will not use the same key while in transmission during the lifetime of the network. Each unique key will be generated by tracking the serial number of PV. Even though the PV is probabilistic and subject to repetition, the serial number is fixed. A frame key will be generated by taking a 48 bit long PV serial number as an input to a key generating function.

The function can be described as follows: The PV serial number is split into six field elements. Each field element is one byte long. Then the bits in the odd field elements are complemented. After this bit reversal, the resultant bit string is shifted left by 600, that is, each field element is shifted left by one byte. The final bit string is the frame key.

It should be noted that this frame key is not directly used for making the cipher. Rather it is used to generate a 256 byte long S-box. The rule for making this box is as follows: first we map the frame key in the first 6 bytes in the S-box. Then we shift the string by 1 bit to the right, filling the next 6 bytes in the box. After 42 cycles are taken where bytes in each successive cycle are dictated by the bytes in the previous cycle, we have 252 bytes mapped in the S-box. For completing the next four bytes, we take the first four bytes in 42nd cycle and take a right shift by 1 bit. The generation of frame key and then the subsequent S-box is given in appendix A.

V. Key Scheduling

In the key scheduling, first the 24 bit long pointing vector (PV) is split into three bytes. Since PV is a random number, the content in the first byte can be all zeros. In order to avoid working with all zeros in this byte, we add 1 to it. After this addition, we label the first byte as I, the second as J, and the third as K. Then the decimal equivalent of J is calculated. If, for example, the decimal equivalent of J is 12, the content in the 12th byte in the S box is picked out, which is Xored with I, and the resultant byte is N. In order to create a 128 bit long key stream, now we trace the byte K from the leftmost bit. If it is 1, each bit in N is complemented and shifted right by I bit. If the bit in K is 0, N remains the same and shifted right by I bit. This key stream can be of any length, but it has to be a multiple of 32. This key schedule is given in appendix B.

VI. Results and Discussion

In Fig. 2, the performance of RC4, AES and our proposed algorithm (TLEP) have been shown in terms of sharing the CPU load. With a small block size, RC4 tends to acquire a far greater CPU time for its processing, and the AES [5] tends to consume much less time and TLEP takes even less. However, RC4 is operating using less CPU processing time and reducing the work load on the CPU when it encrypts large data blocks. Thus, AES and TLEP are suitable for a device which has low processing power (such as wireless devices) to encrypt small size packets.

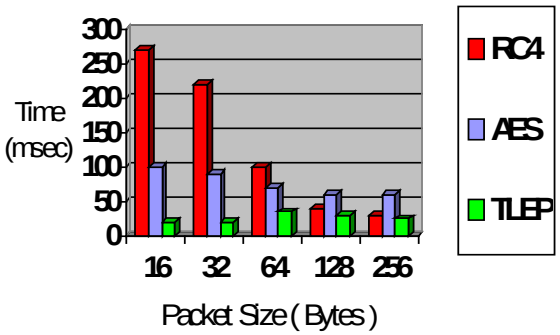

Figure 2. CPU Processing Time of RC4, AES and TLEP.

RC4 is vulnerable mostly because only one key is used to encrypt and decrypt data. The weakness of this algorithm is also attributable to the use of a periodic IV in making the cipher. Taking note of this, we have proposed changing the cipher key for each instance of transmission. This enhancement will effectively guard against the known vulnerabilities now affecting the present system. This idea can also be applied to AES though it is far more robust than RC4. In AES a base cipher key is always fixed, which is fed into 10 subsequent round keys in the key scheduling part [6, 7, 8]. If even the base cipher key is changed for each instance of transmission by the algorithm that we have proposed in this paper, this will add further cryptographic strength to the AES.

The second part of this paper deals with a simplistic but very effective S-box construction. And the third aspect that we dealt with is a secured and flexible key scheduling algorithm. These three aspects of our algorithm make it almost impossible to break a cipher.

VII. Conclusions

In the last few years, WLAN has gained a substantial momentum as it offers fast access of multiple services

inside government offices and institutions. However, WLANs are vulnerable due to the nature of radio broadcasting and the lack of physical connections to the equipment as of wired LAN in order to connect to the network. It is therefore, important to provide appropriate security to WLAN, which ensures the robustness against malicious attacks. WEP is the earlier versions of the IEEE 802.11 wireless LAN standard which is based on the RC4 encryption algorithm, with a secret key of 40 bits or 104 bits being combined with a 24-bit initialization vector to encrypt the plaintext message M and its checksum–the integrity check value (ICV). The initialization vector is the key to WEP security, so as to maintain a fair level of security. The IV should be incremented for each packet so that subsequent packets are encrypted with different keys. The shared key design in WEP requires the network administrator to trust many users with the same authentication credentials for the same set of access points (APs). It is clear that WEP encryption does not provide sufficient wireless network security and can only be used with higher-level encryption solutions (such as VPNs). However, many WiFi devices primarily WLAN adapter cards support the encryption algorithm in hardware. Moreover, old devices support RC4. The aim of this paper is to rectify the drawbacks in the WEP security mechanism. In our paper, we have proposed a new algorithm where a unique cipher key is used for each instance of encryption. This technique will effectively guard against the known vulnerabilities. This paper also concerns about very secured and flexible key scheduling algorithm. These two aspects show an alternative for the existing WEP for WLAN. This will ensure the stringent security of wireless data transmission over WLAN.

REFERENCES

[1] Borisov, N., Goldberg, I. and Wagner, D. (2001) 'Intercepting mobile communications: the insecurity of 802.11', TheSeventh Annual International Conference on Mobile Computing and Networking, (MOBICOM), 2001.

[2] L. M. S. C. of the IEEE Computer Society. 'Wireless LAN medium access control (MAC) and physical layer (PHY) specifications.' IEEE Standard 802.11, 1999 Edition, 1999.

[3] Rivest, R.L. 'The RC4 Encryption Algorithm', RSA Data Security, Inc., March, 1992.

[4] Xiao, Y. Bandela, C. Du, X. Dass, E 'Security mechanisms, attacks and security enhancements for the IEEE 802.11 WLANS' Int. J. Wireless an Mobile Computing. Vol. 1. Nos. ¾. 2006.

[5] P. Prasithsangaree, P. Krishnamurthy, 'Analysis of Energy Consumption of RC4 and AES Algorithms in Wireless LANs'. Globecom, IEEE Conference Paper, August 2003.

[6] Joan Daemen and Vincent Rijmen, AES submission document on Rijndael, Version 2, September 1999.

[7] M.N. Islam, M. Mia, M. Chowdhury, M.A.Matin, 'Effect of Security Increment to Symmetric Data Encryption through AES Methodology' Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008(SNPD '08). 6-8 Aug. 2008, PP.291 – 294, Phuket, Thailand.

[8] M A Matin, M M Hossain, M F Islam, M N Islam and M M Hossain "Performance evaluation of symmetric encryption algorithm in MANET and WLAN" 2009 international conference for TECHPOS, 14-15 Dec. 2009, pp. 1-4, Kuala Lumpur, Malaysia.

AUTHOR'S BIOGRAPHY

M A KABIR is graduated from North South University. His current research interests in security and grid computing.

K A SAYEED is working as an Assistant Professor at North South University (NSU), Bangladesh. He is also a proctor of NSU. He is a physicist and his current research interest includes robotics.

M A MATIN currently works in the Department of Electrical and Electronic Engineering, Institut Teknologi Brunei (ITB), Bandar Seri Begawan, Brunei Darussalam, as an Associate Professor. He has published over 60 refereed journals and conference papers. His current research interests include UWB communication, wireless sensor networks, cognitive radio, EM modeling, and antenna engineering.

T MEHENAZ is graduated from BUET. His research interest is in computing.

M KAMRUZZAMAN is graduated from CUET. His research interest is in the area of security issues.

SOME COMMON TERMS:

WEP	Wired Equivalent Privacy is the original security standard for WLANs.
RC4	It is a stream cipher and is widely used in many applications such as IEEE 802.11 WEP.
AES	Advanced Encryption Standard is a specification for the encryption of electronic data.
WPA	Wi-Fi Protected Access is a subset of 802.11i security standard.
TKIP	Temporal Key Integration Protocol is the part of 802.11i encryption standard.
WPA2	Wi-Fi Protected Access 2 offering more protection to WLANs users.

APPENDIX A: KEY GENERATOR

The generation of frame key and then the subsequent S-box is given in the following C++ code

```
char PVs[48];
char PVs[48];
char STemp[2048];
char s[256][8];
int jInt=0
    for(loop=47;loop>39;loop--) {
      if(PVs[loop]=='0')
         PVs [loop]='1';
      else
         PVs [loop]='0';
     }

   for(loop=31;loop>23;loop--) {
      if(PVs [loop]=='0')
        PVs [loop]='1';
      else
         PVs [loop]='0';
    }

    for(loop=15;loop>7;loop--)  {
      if(PVs [loop]=='0')
        PVs [loop]='1';
      else
     PVs [loop]='0';
    }

   for(loop=0;loop>48;loop++)  {
     printf("%c", PVs [loop]);
    }

for(loop=0;loop<48;loop++) {
     temp2[loop]= PVs [loop];
    }
   for(loop=0;loop<39;loop++){
     PVs [loop]=temp2[loop+8];
}
 jInt=0;
for(loop=40;loop<47;loop++) {
  PVs [loop]=temp2[jInt];
   jInt=jInt+1;
  }
for(loop=0;loop<48;loop++) {
   printf( "%c", PVs [loop]);
  }
jInt=0;
for(loop=0;loop<2016;loop++) {
   STemp[loop]= PVs [jInt];
   if(jInt==47)
                jInt=0;
   else
                jInt=jInt+1;
  }
jInt=0;
for(loop=2016;loop<2048;loop++) {
   STemp[loop]= PVs [jInt];
   jInt=jInt+1;
  }
//printf("\n\n");
int row=-1,col=-1;
for(loop=0;loop<2048;loop++) {
  // printf("%c",STemp[loop]);
 if(loop%8==0) {
 row++;
 col=0;
 }
 else {
   col++;
 }
 s[row][col]=STemp[loop];
  }
```

APPENDIX B: KEY SCHEDULING

This key schedule is given in the following C++ code:

```
char i[8],j[8],k[8];
int jInt=0;
                    for(loop=0;loop<8;loop++)
                       i[loop]=PV[loop];

                       for(loop=8;loop<16;loop++)
                       j[loop-8]=PV[loop];

                       for(loop=16;loop<24;loop++)
                       k[loop-16]=PV[loop];

     char *i1=addone(i);
     for(loop=0;loop<8;loop++) {
                        i[loop]=i1[loop];
      }
    for( loop=0;loop<8;loop++) {
           printf("%c",i[loop]);
     }
     printf("\n     The value of j=");
    for(loop=0;loop<8;loop++) {
           printf("%c",j[loop]);
     }

           int intJ=get int value of J(j);
           char newj[8];
           for(loop=0;loop<8;loop++) {
                    newj[loop]=s[intJ][loop];
     }

           char n[8];
           char *n1=xor(i,newj);
           for(loop=0;loop<=7;loop++) {
                    n[loop]=n1[loop];
            }
            printf("\n   n=((i+1) XOR Sj)=");
            for( loop=0;loop<8;loop++) {
                    printf("%c",n[loop]);
            }

          for(loop=0;loop<8;loop++){
             if(k[loop]=='1') {
                                for(int a=0;a<8;a++){

keystream[loop*16+a]=complement(n[a]);
                                }
                                for(a=8;a<15;a++){

keystream[loop*16+a]=complement(n[a-7]);
                                }

keystream[loop*16+a]=complement(n[0]);
                        }
                else {
                        for(int a=0;a<8;a++){
                         keystream[loop*16+a]=n[a];
                                }
                                for(a=8;a<15;a++){
keystream[loop*16+a]=complement(n[a-7]);
                                }
keystream[loop*16+a]=complement(n[0]);
                }

             }
          printf("\n     The Value of k=");
             for( loop=0;loop<8;loop++) {
              printf("%c",k[loop]);
               }
              for(loop=0;loop<128;loop++){
                printf("%c",keystream[loop]);
```