WEP: The "Wired Equivalent Privacy" Algorithm.

7 November 1994
Revision 1.0

Abstract

This submission proposes an optional "wire equivalent" encryption algorithm to be optionally implemented in 802.11 compliant stations. The algorithm exhibits good security and efficiency, is self-synchronizing and is free of IP restrictions.

Issues Addressed:

6.6 Is there any additional work on Security that needs to be done by 802.11 in addition to the work that is done by 802.10?

6.10 Shall the minimal Security algorithms set be extended to include a Privacy equivalent to wired LANs?

Introduction

With the decision to incorporate 802.10b Secure Data Exchange in the 802.11 standard, we have provided a mechanism that allows cooperating stations to communicate in a secure fashion. However, 802.10b does not afford any security in and of itself; it is up to 802.11 to determine which security services are required on the wireless data link and to specify implementations.

This paper defines "security equivalent to that of a wired LAN" as *at least* protecting authorized users of a wireless LAN from casual eavesdropping and data injection. The first of these LAN security threats is formally known as unauthorized disclosure and can be protected against by the use of a data confidentiality (privacy) service[1]. The second issue is more complex and usually requires the use of security services in addition to privacy (e.g. Authentication).

Eavesdropping is a familiar problem to users of other types of wireless technology. For example, many corporations have policies which prohibit employees from discussing confidential business over cellular telephones. By specifying a wired LAN equivalent data confidentiality service in the 802.11 standard, a significant barrier to market penetration can be eliminated.

[1] IEEE Std 802.10-1992, "Interoperable LAN/MAN Security (SILS), 5 February 1993

Data confidentiality depends on an external key management service to authenticate users and distribute data enciphering/deciphering keys, and on an appropriate cryptographic algorithm. While the security of the cryptosystem may by reduced by a poor choice for either of these components, they are complementary functions and may be considered independently. This submission focuses on a cryptographic algorithm (WEP).

The algorithm described is in the public domain. The algorithm was posted to various Internet news groups during 1994. Subsequent analysis of the algorithm by the Internet security community has indicated that the algorithm is resource efficient, quick, and reasonably secure. The original Internet posting asserted that the algorithm is functionally equivalent to the RC4 algorithm from RSA. For the purpose of 802.11 "wired equivalency" it is irrelevant whether or not the algorithm actually is functionally equivalent to RC4, it might in fact be either better or worse than RC4.

The posted public algorithm does meet the requirements of 802.11 for providing a "wired equivalent" privacy algorithm. For the purposes of this paper, the algorithm is given the name "WEP" for "Wired Equivalent Privacy".

Properties of the WEP Algorithm

WEP was evaluated against the desired properties of a MAC layer cryptographic algorithm as discussed in issue 6.10:

The following paragraphs use term (k and IV) which are defined in the theory of operation section.

Reasonably Strong:
The security afforded by the algorithm relies on the difficulty of discovering the secret key through a brute-force attack. This in turn is related to the length of the secret key (usually expressed in bits) and the frequency of changing keys. However, it may be an easier problem to discover k through statistical methods if the key sequence remains fixed and significant quantities of ciphertext are available to the attacker. WEP avoids this by frequently changing the IV and hence k.

Self Synchronizing:
Provided by the IV, as described. This property is critical for a data-link level encryption algorithm, where "best effort" delivery is assumed and packet loss rates can be high. An algorithm that assumes reliable delivery in order to maintain synchronization between sender and receiver would not provide acceptable performance.

Efficient:
The WEP algorithm is very efficient in comparison to traditional block ciphers. It uses few resources and can be implemented efficiently in either hardware or software. Refer to the algorithm specification section for more details.

IP content:
The WEP proposal in this paper is IP free. A previous similar proposal (RT; 94/22) contained two pieces of IP: The RC4 algorithm licensed by RSA (not used in this proposal); and an implementation specific key caching scheme (absent as it is not required for system operation).

Exportability:
Given the current political and legal climate in the United States regarding Cryptography, it is not possible to predict the exportability of any specific privacy scheme. In fact, export licenses are granted for specific products, not for algorithms. Further, there is no legal guarantee that two different implementations of the

same algorithm will be treated identically for export consideration. (While this may not sit well with some, it is factual as of the writing of this paper.)

Every effort has been made to design the WEP system operation so as to maximize the chances of export via the Commerce Department.

Requirement for an 802.11 Option:
Because of the interest of 802.11 members in making international products, coupled with the vagary of US export law, the usage of the WEP algorithm is specifically proposed to be an optional portion of the 802.11 standard.

Detailed draft text for WEP:

The remainder of this document is written as changes to 20B3 and contains the detailed text and diagrams that are proposed for adoption.

Motion:

That the following detailed changes (derived from draft 20B3) be adopted and incorporated in the 20B4 802.11 draft standard and that Issues 6.6 and 6.10 be closed to reflect the adoption of the WEP proposal.

1.2 Definitions

Wired Equivalent Privacy (WEP). The optional cryptographic privacy algorithm specified by 802.11 used to provide data confidentiality which is subjectively equivalent to a wired media.

1.3 Abbreviations

WEP = Wired Equivalent Privacy.

2.4.3.2 Privacy

IEEE 802.11 specifies an optional privacy algorithm (WEP) that is designed to satisfy the goal of wired LAN "equivalent" privacy. The algorithm is not designed for ultimate security but rather to be "at least as secure as a wire". See section XX for more details.

2.7.5 Privacy

Note: 802.10 does not specify specific cryptographic algorithms for privacy. P802.11 has registered the following algorithms with 802.10:

No Privacy Algorithm in use: Value = ??

Wired Equivalent Privacy (WEP) algorithm: Value = ??

This satisfies the minimal operational needs of 802.11.

Additional privacy algorithms, which have been registered with 802.10 for use within 802.11 implementations, and were known at the time of publication are contained in appendix XX.

2.7.6 Authentication

Note: 802.10 does not specify specific cryptographic algorithms for authentication or privacy. However the algorithm numbers must be known for proper operation of 802.11. P802.11 has registered the following algorithms with 802.10:

No Authentication algorithm in use: Value = ??

This satisfies the minimal operational needs of 802.11.

Additional authentication algorithms which have been registered with 802.10 for use within 802.11 implementations and were known at the time of publication are contained in appendix XX.

x.x.x.x The Wired Equivalent Privacy Algorithm (WEP)

Introduction

Eavesdropping is a familiar problem to users of other types of wireless technology. P802.11 specifies a wired LAN equivalent data confidentiality algorithm. Wired equivalent privacy is defined as protecting

authorized users of a wireless LAN from casual eavesdropping. This service is intended to provide functionality for the Wireless LAN equivalent to that provided by the physical security attributes inherent to a wired media.

Data confidentiality depends on an external key management service to authenticate users and distribute data enciphering/deciphering keys. P802.11 specifically recommends against running an 802.11 with privacy but without authentication. While this combination is possible, it leaves the system open to significant security threats.

Properties of the WEP Algorithm

The WEP algorithm has the following properties:

Reasonably Strong:

The security afforded by the algorithm relies on the difficulty of discovering the secret key through a brute-force attack. This in turn is related to the length of the secret key and the frequency of changing keys. WEP allows for the changing of the key (*k*) and frequent changing the Initialization Vector (IV).

Self Synchronizing:

WEP is self-synchronizing for each message. This property is critical for a data-link level encryption algorithm, where "best effort" delivery is assumed and packet loss rates can be high.

Efficient:

The WEP algorithm is efficient and can be implemented in either hardware or software.

Exportability:

Every effort has been made to design the WEP system operation so as to maximize the chances of export via the Commerce Department. However, due to the legal and political climate toward Cryptography at the time of publication, no guarantee could be made that any specific 802.11 implementations which uses WEP would be exportable from the United States.

Therefore, the implementation and use of WEP is an 802.11 option.

WEP Theory of Operation

The process of disguising (binary) data in order to hide its information content is called **encryption**[2]. Data that is not enciphered is called **plaintext** (denoted by *P*) and data that is enciphered is called **ciphertext** (denoted by *C*). The process of turning ciphertext back into plaintext is called **decryption**. A **cryptographic algorithm**, or cipher, is a mathematical function used for enciphering or deciphering data. Modern cryptographic algorithms use a key (denoted by *k*) to modify their output. The encryption function *E* operates on *P* to produce *C:*

$E_k(P) = C$

In the reverse process, the decryption function *D* operates on *C* to produce *P:*

[2]Bruce Schneier, "Applied Cryptography, Protocols, Algorithms and Source Code in C", John Wiley & Sons, Inc. 1994

$D_k(C) = P$

As illustrated in Figure 1, note that if the same key is used for encryption and decryption then

$D_k(E_k(P)) = P$

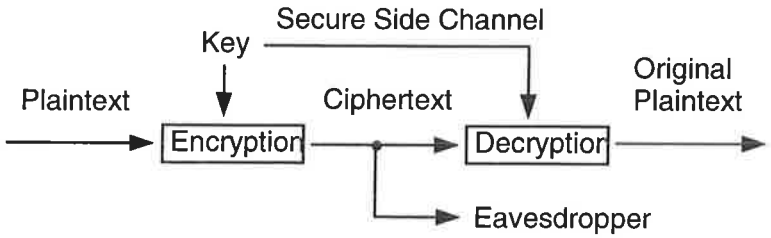

Figure 1. A Confidential Data Channel

The WEP algorithm proposed in this submission is a form of electronic code book in which a block of plaintext is bitwise XOR'd with a pseudo random key sequence of equal length. The key sequence is generated by the WEP algorithm.

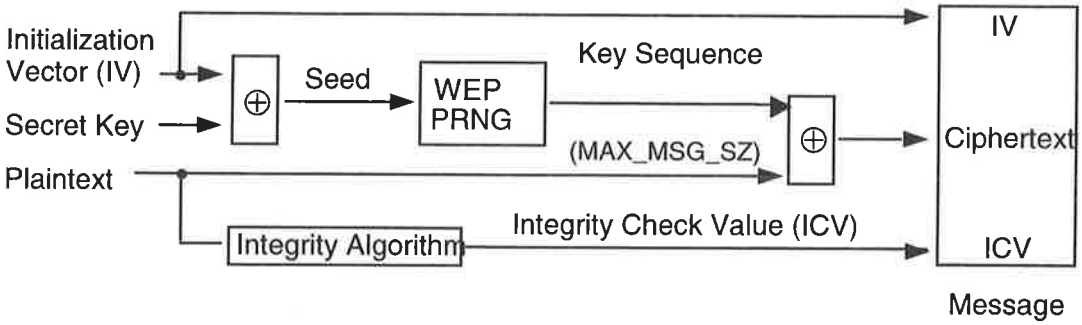

Figure 2. WEP Encipherment Block Diagram

Referring to Figure 2 and following from left to right, encipherment begins with a **secret key** that has been distributed to cooperating stations by an external key management service. WEP is a symmetric algorithm in which the same key is used for encipherment and decipherment.

The secret key is combined with an **initialization vector** (IV) and the resulting **seed** is input to a **pseudo random number generator** (PRNG). The PRNG outputs a **key sequence** k of pseudo-random bits equal in length to the largest possible MSDU. Two processes are applied to the plaintext MSDU. To protect against unauthorized data modification, an integrity algorithm operates on P to produce an **integrity check value** (ICV). Encipherment is then accomplished by mathematically combining the key sequence with P. The output of the process is a **message** containing the resulting ciphertext, the IV, and the ICV.

The WEP PRNG is the critical component of this process, since it transforms a relatively short secret key into an arbitrarily long key sequence. This greatly simplifies the task of key distribution as only the secret key needs to be communicated between stations. The IV extends the useful lifetime of the secret key and provides the self-synchronous property of the algorithm. The secret key remains constant while the IV changes periodically. Each new IV results in a new seed and key sequence, thus there is a one-to-one correspondence between the IV and k. The IV may be changed as frequently as every MSDU and, since it travels with the message, the receiver will always be able to decipher any message. The IV may be transmitted in the clear since it does not provide an attacker with any information about the secret key.

The WEP key (k) is 40 bits.

Because IV and the ICV must be transmitted with the MSDU, fragmentation may be invoked. The WEP algorithm is applied to an MSDU. The {IV, MSDU, ICV} triplet forms the actual data to be sent in the data frame.

For WEP protected Data frames, the first octets of the frame contain the IV for the MSDU. The WEP IV is 16 bits. The IV is followed by the MSDU, which is followed by the ICV. The WEP ICV is 32 bits. The WEP Integrity Check algorithm is CRC 32.

The entire {IV, MSDU, ICV} package may be split into several fragments (depending on the realtive values of the MSDU and the active MPDU size).

As stated previously, WEP combines k with P using bitwise XOR.

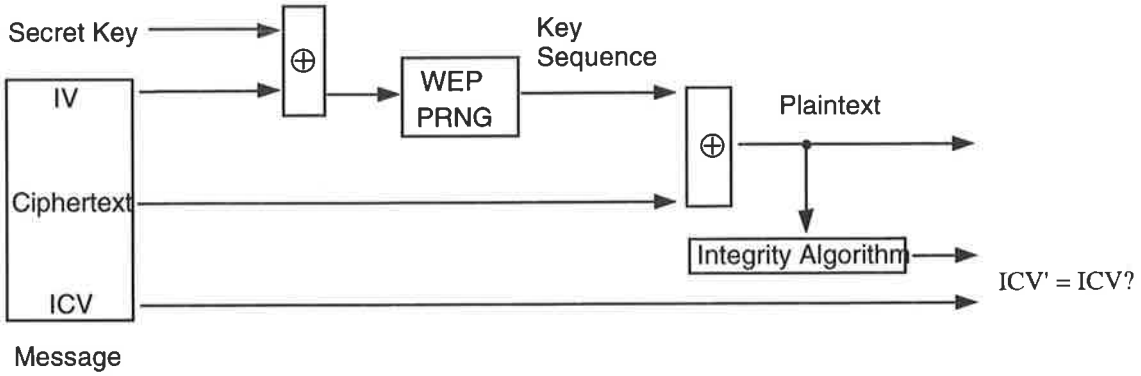

Figure 3. WEP Decipherment Block Diagram

Referring to Figure 3 and following from left to right, decipherment begins with the arrival of a message. The IV of the incoming message is used to generate the key sequence necessary to decipher the incoming message. Combining the ciphertext with the proper key sequence yields the original plaintext. If desired, this may be verified by performing the integrity algorithm on the recovered plaintext and comparing the output ICV' to the ICV transmitted with the message.

WEP Algorithm specification:

The WEP algorithm is specified by the following C code fragments.

```
/* wep.h */
typedef struct wep_key
{
    unsigned char state[256];
    unsigned char x;
    unsigned char y;
} wep_key;

void prepare_key(unsigned char *key_data_ptr,int key_data_len, wep_key *key);
void wep(unsigned char *buffer_ptr,int buffer_len,wep_key * key);


/*wep.c */
#include "wep.h"
static void swap_byte(unsigned char *a, unsigned char *b);
void prepare_key(unsigned char *key_data_ptr, int key_data_len, wep_key *key)
{
    unsigned char swapByte;
    unsigned char index1;
    unsigned char index2;
    unsigned char* state;
    short counter;

    state = &key->state[0];
    for(counter = 0; counter < 256; counter++)
    state[counter] = counter;
    key->x = 0;
    key->y = 0;
    index1 = 0;
    index2 = 0;
    for(counter = 0; counter < 256; counter++)
    {
        index2 = (key_data_ptr[index1] + state[counter] + index2) % 256;
        swap_byte(&state[counter], &state[index2]);
        index1 = (index1 + 1) % key_data_len;
    }
}

void wep(unsigned char *buffer_ptr, int buffer_len, wep_key *key)
{
    unsigned char x;
    unsigned char y;
    unsigned char* state;
    unsigned char xorIndex;
    short counter;

    x = key->x;
    y = key->y;
```

```
    state = &key->state[0];
    for(counter = 0; counter < buffer_len; counter ++)
    {
        x = (x + 1) % 256;
        y = (state[x] + y) % 256;
        swap_byte(&state[x], &state[y]);

        xorIndex = state[x] + (state[y]) % 256;

        buffer_ptr[counter] ^= state[xorIndex];
     }
    key->x = x;
    key->y = y;
}

static void swap_byte(unsigned char *a, unsigned char *b)
{
    unsigned char swapByte;

    swapByte = *a;
    *a = *b;
    *b = swapByte;
}
```