IEEE 802.11
Wireless LAN Medium Access Control and Physical Layer Specifications

The RT Data Confidentiality Algorithm

7 March1994

Kerry Lynn
Apple Computer, Inc.
One Infinite Loop, MS 301-4J
Cupertino, CA 95014
email: lynn@applelink.apple.com

Abstract

This submission proposes that a standard data confidentiality algorithm be implemented in all 802.11 stations that provide security services. A candidate algorithm is described which exhibits adequate security and efficiency, is self-synchronizing, and has been approved for export under jurisdiction of the US Department of Commerce.

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.

A recent submission from ETSI [1] defines "security comparable 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 protectected against by the use of a data confidentiality (privacy) service [2]. The second issue is more complex and requires the addition of an integrity service.

Eavesdropping is a familiar problem to users of other wireless technologies. For example, many corporations have policies that prohibit employees from discussing confidential business over cellular telephones. Apple believes that by including a basic data confidentiality service in the 802.11 standard, a significant barrier to market penetration can be eliminated.

Data confidentiality depends on 1) an external key management service to authenticate users and distribute data enciphering/deciphering keys, and 2) an appropriate confidentiality 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 confidentiality algorithm (RT) and proposes that it be included as a basic feature in all 802.11 stations that provide security services.

Theory of Operation

The process of disguising (binary) data in order to hide its information content is called encryption or **encipherment** [3]. 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 or **decipherment**. 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:

$$D_k(C) = P$$

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

$$D_k(E_k(P)) = P$$

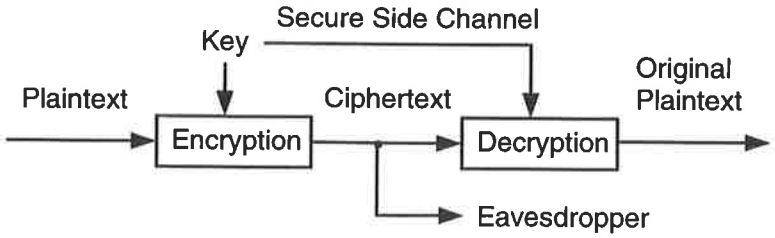

Figure 1. A Confidential Data Channel

The algorithm proposed in this submission is a form of electronic codebook in which a block of plaintext is bitwise XOR'd with a pseudorandom key sequence of equal length. The key sequence is generated by the proprietary RC4 algorithm licensed from RSA Data Security, Inc.

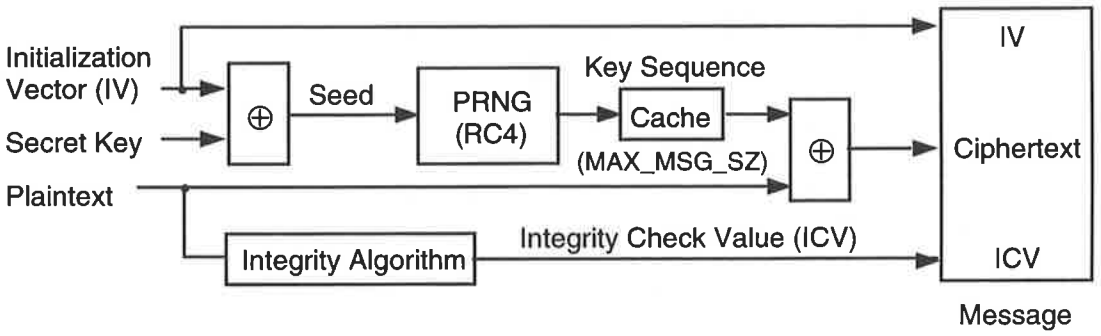

Figure 2. RT Encipherment Block Diagram

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. RT 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 **pseudorandom 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 PRNG is the critical component of this system, 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-syncronous 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 transmitted in the clear since it does not provide an attacker with any information about the secret key.

As stated previously, RT combines *k* with *P* using bitwise XOR; this combination step is much less complex than generating the key sequence. In a software implementation, the key sequence

may be cached so that it can be used to encipher more than one plaintext MSDU. In this way, the "cost" of computing the key sequence may be spread over several MSDUs although this lessens the security somewhat due to the reuse of *k*.

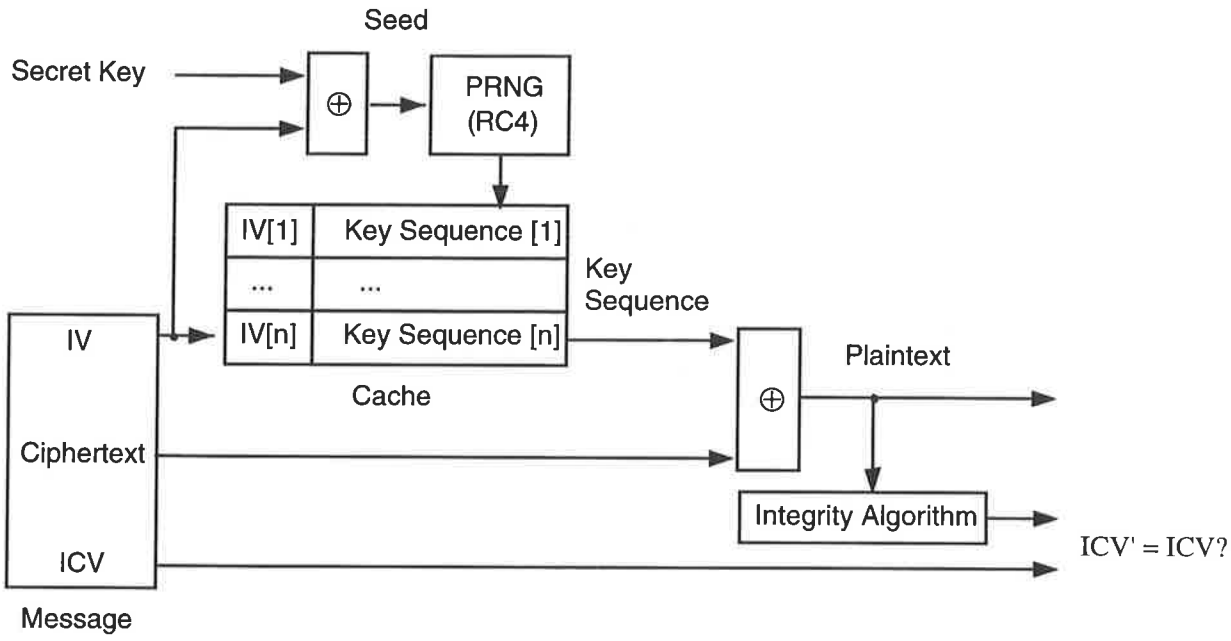

Figure 3. RT Decipherment Block Diagram

Referring to Figure 3 and following from left to right, decipherment begins with the arrival of a message. The IV of the incoming message may be used to search an optional cache of <IV, key sequence> pairs; if the IV is found then the corresponding key sequence has already been generated and the algorithm proceeds to the combination step. If the IV was not located, the key sequence necessary to decipher the incoming message is generated and (optionally) stored in the cache for future use. Combining the ciphertext with the proper key sequence yeilds 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.

Properties of the Algorithm

RT should be evaluated against the desired properties of a MAC layer confidentiality algorithm as discussed in issue 6.10:

Strong: The security afforded by the algorithm should rely 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. RT avoids this by frequently changing the IV and hence *k*.

It should be noted that the goal of RT is to provide the *minimum* acceptable level of security. Additional security may be provided by higher layer protocols.

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

Efficient: RT is efficient in two ways: it is based on a stream cipher that is extremely efficeient even in software implementation, and it is unique in that the "cost" of the most computationally intensive part of the algorithm may be amortized over several packets.

Exportable/Importable: While Apple has obtained an export license for the described algorithm, this is unfortunately not a guarantee that all parties will be granted a license. Nevertheless, there are two things that suggest this will be the case: the precedent has been set, and the algorithm strictly conforms to the agreement worked out between the State Department and the Software Publisher's Association. A copy of "Procedure for Submitting a Commodity Jurisdiction Request for a Mass Market Software Product that Contains Encryption" may be obtained by writing to:

United States Department of State
Department of Politico-Military Affairs
Office of Defense Trade Controls
Washington, D.C. 20522-0602

Licensable:

Itellectual Property Statements

" Elements of this proposal may be protected by one or more issued or pending patents owned by Apple Computer, Inc. In the event that this proposal is adopted into the IEEE 802.11 Standard, and the Standard cannot be practiced without the use of such patent(s), Apple agrees upon request to grant a non-exclusive license under such patent(s) on a nondiscriminatory basis and on reasonable terms and conditions, provided a similar grant under licensee's patents within the scope of the license granted to licensee is made available upon request to Apple."

"Elements of this proposal are trade secrets of RSA Data Security, Inc. In the event that this proposal is adopted into the IEEE 802.11 Standard, RSA agrees upon request to grant a non-exclusive license to its technology on a nondiscriminatory basis and on reasonable terms and conditions."

A license for the RT algorithm may be obtained by contacting:

RSA Data Security, Inc.

100 Marine Parkway
Redwood City, CA 94065

Conclusion

It is unlikely that a large corporate customer considering a volume purchase of 802.11 RF LAN adapters would do so with the knowledge that they would be vulnerable to unauthorized LAN access or disclosure of sensitive information. It is also unlikely that they would be willing to retrofit security services into their existing wired infrastructure. The solution is to provide security services at the point of vulnerability - the wireless data link. It follows that in order to achieve interoperability in a secure environment, one or more algorithms must be shared by all stations that support security services. This submission proposes a canditate algorithm that strikes a balance between the conflicting contstraints of strength, efficiency, and exportability.

Appendix A: Implementation Specification

This section gives a pseudo-code description of the implementation that conforms to the 802.10 approach adopted by the committee. This section should be included in the IEEE 802.11 Specification as an aid to interoperability.

```
/*  D E F I N E S  */

#define KEY_SZ 8
#define MAX_MSG_SZ 1500

typedef unsigned char uchar;
typedef uchar Key[KEY_SZ];
typedef uchar Sequence[MAX_MSG_SZ];
typedef struct Message {
    Key iv;
    Sequence data;
} Message;


/*  C O M M O N   V A R I A B L E S  */

/*
 *  The global variable 'secret_key' is set by a station management function
 *  as a result of an authenticated key distribution operation.  The receiving
 *  station must also have it before any secure messages can be exchanged.
 */
extern Key secret_key;


/*  T X   V A R I A B L E S  */

/*
 *  The (non-negative) global variable 'max_pkts_per_iv' is set by a station
 *  management function according to policy.  If it is low, security and
 *  overhead are increased.  If it is high, security and overhead are
 *  decreased.
 */
extern int max_pkts_per_iv;

Message outgoing;               /* contains the secure tx message         */

static int initialized = 0;     /* non-zero iff following params init'd   */
static int pkt_count;           /* number of packets sent with current iv */
static Key tx_iv;               /* transmit initialization vector         */
static Key tx_seed;             /* seed to PRN generator                  */
static Sequence tx_key_seq;     /* transmit pseudorandom key sequence     */


/*  R X   V A R I A B L E S  */

/*
 *  Rx has separate variables, since rx and tx are concurrent processes
 */
static Key rx_iv;               /* receive initialization vector          */
static Key rx_seed;             /* seed to PRN generator                  */
static Sequence rx_key_seq;     /* receive pseudorandom key sequence      */

static Message cache_entry;

/*
```

```
 *  The receiving station keeps a cache of recently seen <iv, sequence>
 *  tuples.  If the iv of an incoming message matches an entry in the
 *  cache, the corresponding PRN sequence has already been generated.
 *  All that need be done to decrypt the ciphertext is an relatively
 *  inexpensive Combine() operation.
 */
static Message *Cache;


/*  C O M M O N   F U N C T I O N S  */

/*
 *  'NewIV' generates a pseudorandom initialization vector and stores it in
 *  'iv'.  Ideally, each iv is randomly and uniformly distributed across the
 *  range of possible values.
 */
extern void NewIV(Key *iv);

/*
 *  'NewSequence' generates a maximum MSDU length pseudorandom sequence and
 *  stores it in 'buffer'.  Ideally, each unique 'seed' results in a unique
 *  sequence without cycles, etc.
 */
extern void NewSequence(Key *seed, Sequence *buffer);

/*
 *  'Combine' mathematically combines 'object1' and 'object2' storing the
 *  result in 'object3'.  'objectSz' is the length of the objects in bytes.
 *  Ideally, the cost of this operation is much less than generating a new
 *  key sequence.  In this example, the operation is a bitwise XOR.
 */
static void
Combine(uchar *object1, uchar *object2, uchar *object3, int objectSz)
{
    int i;

    for (i = 0; i < objectSz; i++)
        object3[i] = object1[i] ^ object2[i];
}


/*  T X  F U N C T I O N S  */

/*
 *  'InitVars' sets up the local parameters required to generate a secure
 *  message.
 */
static void
InitVars()
{
    pkt_count = 0;
    NewIV(&tx_iv);
    Combine(secret_key, tx_iv, tx_seed, KEY_SZ);
    NewSequence(&tx_seed, &tx_key_seq);
}

/*
 *  'SendMessage' generates a secure message.  The result is placed in the
 *  global parameter 'outgoing'.  Note that as a side effect this function
 *  generates a new key sequence at the rate defined by the global variable
 *  'max_pkts_per_iv'.
 */
void
SendMessage(uchar *plaintext, int dataSz)
{
```

```
    if (initialized == 0  ||  pkt_count > max_pkts_per_iv) {
        InitVars();
        initialized = 1;
    }
    memcpy(outgoing.iv, tx_iv, KEY_SZ);
    Combine(tx_key_seq, plaintext, outgoing.data, dataSz);
}


/*  R X   F U N C T I O N S  */

/*
 *  'GetCache' searches the receiving station's cache for an entry matching
 *  the input 'message.iv'.  Iff the entry is found, the return value is
 *  non-zero and the corresponding PRN sequence is returned in 'message.data'.
 */
extern int
GetCache(Message *message);

/*
 *  'SetCache' enters a new <iv, sequence> tuple into the receiving station's
 *  cache.
 */
extern void
SetCache(Message *message);

/*
    This function sets up the local parameters required to decrypt a secure
    message.
*/
static void
NewEntry(Message *cache_entry)
{
    Key seed;            /* seed to PRN generator */

    Combine(secret_key, (uchar *) cache_entry->iv, seed, KEY_SZ);
    NewSequence(&seed, (Sequence *) cache_entry->data);
}

/*
    This function decrypts a secure message.
*/
void
RecvMessage(Message *incoming, int dataSz)
{
    /* use the incoming message iv as key for the cache search */
    memcpy(cache_entry.iv, incoming->iv, KEY_SZ);
    if (GetCache(&cache_entry) == 0) {
        NewEntry(&cache_entry);
        SetCache(&cache_entry);
    }
    /* combine the PRN sequence with the data, decrypting it in place */
    Combine(cache_entry.data, incoming->data, incoming->data, dataSz);
}
```

References

[1] IEEE P802.11-94/9, "Radio Equipment and Systems (RES); HIPERLAN Security Information (input for STAG)", 14 December 1993

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

[3] W. Ford, *Computer Communications Security*, Prentice-Hall, 1994