IEEE P802.11

Wireless Access Method and Physical Layer Specification

Bumps On The Road To Privacy

Michael Fischer
Digital Ocean, Inc.
4242–3 Medical Drive
San Antonio, TX 78229
Telephone: +1–210–614–4096
Facsimile: +1–210–614–8192
email: mfischer@CHILD.com

Abstract

At the July, 1995 meeting, the WEP function was changed from operating on each MSDU to operating independently on each MPDU, even when those MPDUs are fragments of the same MSDU. This change was intended to simplify the data handling needed to implement WEP, without compromising privacy. While this simplification appears justifiable in an abstract view, closer examination, in the context of the actual encryption algorithm, shows **a decrease** in efficiency, and **an increase** in data handling complexity, resulting from these changes. In addition, there is some weakening of the privacy, although probably not enough to drop below the "wired equivalence" level. This submission identifies these problems and recommends reverting to per–MSDU encryption as the most expedient solution that does not compromise the exportability objective.

In addition, this submission identifies two features which are important to achieving the privacy objectives in a real–world environment. One of these features appears to be present, but inadequately specified, in the D2.0 MAC, while the other is absent but simple to add.

The authentication mechanism is not an integral part of WEP, but implementing privacy without authentication is exceedingly difficult. Accordingly, this submission also identifies a serious conflict between the authentication mechanism and the reassociation mechanism.

Category: "things that are broken"

This submission identifies some unanticipated implications of recent changes to the WEP and authentication mechanisms that increase complexity and reduce efficiency. These are not "fatal" problems, since the MAC *could* be implemented with these functions operating as currently specified. However, this author believes strongly that if an optional feature, such as WEP, requires unjustifiable complexity and/or impaired efficiency, most implementers will omit the option. Since the reason for including WEP and authentication in the 802.11 MAC was to make a limited privacy function widely available without requiring a full 802.11 SDE sub–layer, avoidable barriers to widespread implementation of these privacy features need to be avoided.

Introduction

The 802.11 MAC includes authentication and privacy functions in order to compensate for some of the inherently less–secure aspects of the wireless medium. Authentication attempts to fill the role performed on wired media by the limited and controllable set of physical sites at which attachment to the medium is possible. Privacy (WEP) attempts to fill the role performed inherently by the wired media, which contain the vast majority of the transmitted signal within the network cables. These security facilities are optional parts of the 802.11 MAC, due both to the existence of applications that do not require security, and to concerns about export approval of products which use the WEP encryption algorithm.

NOTE: To be precise, authentication is a mandatory function and WEP is an optional function. However, "shared key" authentication uses the WEP encryption mechanism in two places, so a MAC implementation that does not provide the WEP option is implicitly limited to using "open system" authentication. A side effect is that a station that lacks a WEP encryption implementation is unable to associate with a BSS that uses shared key authentication, even if that BSS is not using WEP on data frames.

Several aspects of security in the D2.0 draft create avoidable problems for those implementing (shared key) authentication and WEP. These include:

- WEP encryption is done for each MPDU, which, upon detailed consideration, is **less** efficient and **more** complex than the per–MSDU encryption specified in the D1.2 draft.
- Reassociation is rendered essentially useless for BSS–transition mobility by the requirement that the station requesting reassociation already be authenticated and associated with the new BSS.
- There is not a sufficient mechanism to configure an AP such that unencrypted frames from associated stations are not propagated to the distribution system medium. While this permissive use of WEP is useful in certain cases, many of the users most concerned with the privacy function want to protect their wired infrastructure from such non–private communication.
- While key management is outside the scope of 802.11, a mechanism which must be within the MAC to be useful is missing from WEP. This precludes using several of the most common key management techniques in the absence of a full 802.10 SDE sub–layer.

Each of these problems, along with a recommended solution, is discussed in greater detail below.

Problem #1 — Per–MPDU Encryption Using RC4

At the July, 1995 meeting, the WEP function was changed from encrypting and decrypting entire MSDUs, prior to fragmentation (on transmit) and after reassembly (on receive), to encrypting individual MPDUs, even when those MPDUs are fragments of the same MSDU. This change was intended to simplify the data handling needed to implement WEP, without compromising privacy. While this simplification appears justifiable in an abstract view, closer examination, in the context of the actual encryption algorithm, shows **a decrease** in efficiency, and **an increase** in data handling complexity, resulting from these changes. In addition, there is some weakening of the privacy, although probably not enough to drop below the "wired equivalence" level.

The root of this problem is that several characteristics of the RC4 pseudo–random number generator (PRNG) are quite different from what are generally assumed about PRNGs. All PRNGs produce a sequence of pseudo–random values, designated (R_0, R_1, R_2, ...), based on successive values of their internal generator state (GS_0, GS_1, GS_2, ...). At each invocation, the PRNG evaluates two functions:

$$R_n := F1(GS_n)$$
$$GS_{n+1} := F2(GS_n)$$

The initial generator state is set by applying an initialization function to the seed value:

$$GS_0 := F0(SEED)$$

For a typical PRNG, the size of GS is "small" (<16 bytes) and the initialization function is an identity (GS_0 := SEED). For RC4, the size of GS is "large" (about 258 bytes), and the initialization function involves substantial data manipulation. The relative degree of initialization complexity can be measured by the number of memory accesses into the GS array. For a direct software implementation, this is on the order of 1280 memory operations, although reduction, to 1024 operations should be possible with certain data paths and instruction sets. The initialization operations are not separable by key subset, meaning that the initialization function cannot begin until the entire seed is available, which is at the end of receipt of the initialization vector (IV) field for a station receiving a WEP frame with a new IV value.

The principal benefit from the change to per–MPDU encryption was supposed to be facilitation of "on–the–fly" decryption by making the IV and ICV available in every fragment. Unfortunately, only 1 octet time is available between the end of the IV field and the start of the encrypted MPDU payload, so on–the–fly decryption requires that execution of the initialization function be completed in less than 4 microseconds (for a 2Mbps PHY, 8 microseconds for a 1Mbps PHY). Even if we postulate a data path and instruction set which permits ideal pipelining, in which no memory cycles are lost for instruction fetch, iteration, or other (interrupt) activity, and assume that data memory operations can occur at a sustained 33MHz rate (and ignore the implications this processing rate has for power consumption), the initialization function requires more than 38 microseconds to execute. In practice, far more time (≈100 microseconds) will be needed to execute the initialization function with the common instruction sets for embedded control (i960, 683xx, x86, ARM, SPARC, MIPS, PPC 4xx, etc.), and this does not assume reduced clock rates appropriate for battery operated equipment. Therefore, software–based on–the–fly decryption is not possible. It is also questionable whether a dedicated hardware solution with reasonable cost and power consumption can offer initialization times much below 20 microseconds, but the point is moot as long as the standard says "The WEP algorithm is efficient and can be implemented in either hardware or software" (Section 5.2.2 of D2.0).

If on–the–fly decryption is not done, applying WEP to individual MPDUs rather than MSDUs is a disadvantage whenever fragmentation is used. The presence of per–MPDU encryption with fragmentation decreases efficiency and increases data handling complexity when compared with per–MSDU encryption:

a) Per–MPDU encryption places great importance on changing the IV for every MPDU. (This is discussed in greater detail in the "Related Issue" below.) Take the case of a 1200–octet MSDU transmitted as three, 400–octet fragments. If encrypted as an MSDU, decryption requires one execution of the initialization function, and a (relative) total of 7280 memory operations into the GS array. If encrypted as MPDUs, each with different IV values, decryption requires three executions of the initialization function, and a (relative) total of 9840 memory operations into the GS array — a 35% overhead penalty for per–MPDU encryption!

b) Per–MPDU encryption generates an ICV with each fragment. This increases data handling complexity because the fragment payloads cannot be reassembled simply by concatenating the frame bodies of each received fragment. Rather than improving the "integrity check" function, these extra ICVs actually **reduce** the strength of the integrity check, as is discussed below.

RELATED ISSUE — The Risks of Using the Same IV for Successive Frames

WEP ciphertext is the XOR of the payload plaintext with an equal number of octets generated by the PRNG. Therefore, if the contents of the plaintext are known, or can be accurately predicted, an unauthorized party receiving the ciphertext can determine easily the octet string used for encryption. Because this encrypt string is the output of the PRNG when seeded with a specific (key, IV) pair, this string is only useful for defeating the privacy function when the same IV is used for multiple transmissions over a short period of time (or the attacker is willing to listen long enough to accumulate the 38.65GB of possible decrypt strings for a single key — and the user cooperates by leaving the key unchanged for that long).

There are instances where the contents of large MSDUs are known — the most common is probably the initial frames of a print job being sent to a PostScript® printer. However, the nearly universal instance where some payload contents are known or easily predicted are the intermediate–layer protocol headers at the beginning of each MSDU. The typical risk of reusing an IV is greater for per–MPDU than for per–MSDU encryption:

- With per–MSDU encryption, in the common case where the intermediate–layer protocol headers are the only source of known plaintext, these protocol headers reveal the encrypt string values used to encrypt the next set of protocol headers, whose contents are already known.
- With per–MPDU encryption, under the same conditions, the protocol header in the first fragment of an MSDU reveals the encrypt string values that are used to encrypt the first N (typically 20–80) octets of the subsequent fragments of the MSDU — which contain higher–layer (user) data.

IV re–use, which is the most direct means of mitigating the extra initialization overhead of per–MPDU encryption, degrades the privacy value of per–MPDU encryption much more than of per–MSDU encryption. Of course, the best policy when privacy matters is to change the IV before encrypting each MxDU.

RELATED ISSUE — The Scope of ICV Coverage

The integrity check value (ICV) is used to validate that the correct key has been used to decrypt a received WEP frame. With per–MSDU encryption, the ICV also provides assurance against fragment substitution within an MSDU. With per–MPDU encryption, there is no such end–to–end checking of the MSDU. This is, at best, a small security hole, because the rules for reassembly will operate to discard an MSDU from which a fragment has been removed or inserted. The only risk is fragment substitution, where another member of the BSS (using shared–key encryption) sends an altered fragment in such a manner that the recipient captures that fragment instead of the simultaneous correct fragment. For most receiver signal capture characteristics and fragmentation methodologies, there is greater likelihood of the fragment (hence the MSDU) being discarded due to a CRC error when this is attempted. However, the use of either per–MSDU ICV checking, or unique key values for each station, can fully prevent this risk.

RELATED ISSUE — A software–friendly ICV Algorithm

Another ICV–related issue is that the ICV algorithm must combine data from all covered octets, scattering this data approximately uniformly across the full ICV field. However, the integrity checking by the ICV is to detect flaws such as incorrect decrypt keys or substituted fragments, not missing data due to burst errors on a serial channel. Therefore, CRC–32, whose primary strength is an ability to detect any burst error up to 31 bits long, is not necessarily the best ICV algorithm.

Generation of the CRC–32 polynomial is quite inefficient in software using a general–purpose instruction set. The CRC–32 hardware which is likely to exist on all 802.11 implementations for generating and checking the MAC CRC–32 is probably useless for ICV generation and checking. The MAC CRC–32 must be checked in real time during reception, because the validity of the MAC CRC–32 must be determined in time to send an acknowledgment one SIFS interval after the end of a successful reception. The problems with PRNG initialization preclude real–time decryption, so off line software decryption is likely to be done by a separate execution thread. This thread is unlikely to be able to share the MAC CRC–32 hardware because another reception might commence while ICV checking is in progress. Therefore, using CRC–32 as the ICV algorithm requires software WEP implementations to be even less efficient than listed above, and forces hardware WEP implementations to have a second CRC–32 generator/checker. A better alternative is to use SLRC–3 (LRC with 3–bit circular left shift between octets) to calculate ICV. SLRC–3 has the advantage of being simple and efficient to implement in software or in serial or byte–parallel hardware (serial LRC can be treated as CRC with a polynomial of $x^{32}+1$). The scattering of information content across the multi–octet check field is approximately equal for a CRC and an SLRC (with relatively prime inter–octet shift). It may also be worth considering whether a 16–bit ICV is sufficient. The risk of an erroneous indication of successful decryption is 0.0015% (1.5e–5) with a 16–bit ICV, versus 0.000000023% (2.3e–10) with a 32–bit ICV.

RECOMMENDATION: The best general alternative is to revert to the per–MSDU encryption, as defined prior to the adoption of the motion associated with document 95/138. Changing to a PRNG which has a simpler initialization function is a not a good idea because RC4 with a key length ≤40 bits allows the possibility of expedited export approval. A further improvement, which further facilitates software implementation, is to use SLRC–3 as the ICV algorithm.

Problem #2 — Authentication as a Barrier to Reassociation

The authentication mechanism is not an integral part of WEP, but implementing (useful) privacy without authentication is exceedingly difficult. Unfortunately, there is a conflict between the current definition of the authentication mechanism and reassociation. Reassociation Request and Response frames are class 3 frames (Section 2.5 of D2.0), therefore only permitted when the stations involved are both authenticated and associated. This is always the case for a station reassociating with the same AP for the purpose of changing per–association settings. However, for a station attempting a BSS–transition reassociation, either the distribution system must propagate all known authentication and association information between APs of the ESS, or the mobile station must associate with the new AP prior to reassociating (which violates the current rule that a station may be associated with no more than one AP of an ESS at any time).

The propagation of known authentication and association data among all APs of the ESS is feasible, but would only be of severely limited utility unless an interoperable protocol for inter–AP communication via the distribution system were defined as an exposed interface. Furthermore, using the distribution system in this manner places constraints on the reliability and security of the distribution system that are outside the scope of the current standard and outside the charter of 802.11. A simpler solution is to make Reassociation Request a class 2 frame, while leaving (successful) Reassociation Response as a class 3 frame. This requires the station attempting a BSS–transition reassociation only to authenticate with the new AP before sending the Reassociation Request. By doing the authentication directly with the new AP, there is no need to secure the distribution system medium for propagating authentication data. The new AP must query the old AP (whose MAC address is provided in a field of the Reassociation Request frame) to determine whether the requester is currently associated. Since these requests only take place between APs (which are pre–authenticated to each other as part of the ESS), and only on behalf of authenticated stations (presumably using shared–key authentication), the overall security constraints on the DSM are significantly reduced. To remove the reliability constraints on the DSM, a 3–way transaction can be used for the inter–AP query:

1) Reassociation_Query (newAP, oldAP, StationAddr) — sent to validate Reassociation Request
2) Reassociation_Reply (oldAP, newAP, StationAddr, result) — result indicates current association state
3) Reassociation_Confirm (newAP, oldAP, StationAddr) — confirms result and transfers association to newAP

A security token (generated by the WEP PRNG) could be used if the integrity of the DSM is in doubt.

If the time required for authentication and reassociation (6 frames between station and AP via WM and 3 between new AP and old AP via DSM) is felt to be excessive, two new management frames could be defined to piggyback the reassociation on the final two frames of the authentication handshake. In the case of the 4–way shared key authentication, the reassociation request would be sent in frame 3 along with the challenge response, and the reassociation reply would be sent in frame 4 along with the authentication result.

RECOMMENDATION: Redefine Reassociation Request as a Class 2 frame, define the inter–AP messages initiated by the newAP to the oldAP upon receipt of the Reassociation Request and prior to sending the Reassociation Response, and allow the Reassociation Request/Response to be combined with the final two frames of the Authentication exchange.

Problem #3 — Inadequate Protection of the Infrastructure

The privacy MIB variables allow several of the most important privacy functions, including WEP on/off, a default key, and address–based key mapping. However, there is a missing function that is very important for use at access points: A setting that prevents acceptance of non–WEP data frames. Presently, this function can only be provided if WEP_Key_Mapping is in use and the key mapping table contains entries for all of the possible associated stations. For BSSes which use the Default_WEP_Key, or where the full list of possible stations cannot be easily maintained at each AP, there may still be a need to enforce shared–key privacy. A Boolean MIB entry for "Exclude_Unencrypted_MSDUs" would provide the missing function. This variable should default to True, but the value of this variable is only relevant when WEP_Default is True or WEP_Key_Mapping is active.

RECOMMENDATION: Add the Exclude_Unencrypted_MSDUs MIB variable and related functionality listed above.

Problem #4 — An Unnecessary Obstacle to Key Management

Key management is outside the scope of 802.11. However, several common key management techniques require some cooperation from the frame transfer facility (in this case the 802.11 MAC) to synchronize key changes. The typical form of this cooperation is a "key identifier" in the unencrypted header of each frame with an encrypted payload. The key identifier allows selection among a small number of keys. The most important use for the KeyID is to identify the key used to encrypt each frame when data transfers are allowed to continue, uninterrupted, while a key update transaction is in progress. Another possible use is to select an alternate key for management and/or key exchange transactions.

An example of a WLAN protocol which provides this mechanism is HIPERLAN, which has a 2–bit key identifier. The HIPERLAN identifier allows selection of 3 keys, with the 00 state indicating no key. 802.11 has a superior mechanism to indicate no key — the WEP bit in the frame control field. This is superior because the IV and ICV fields are omitted when WEP=0.

A simple way to add key identifiers to WEP frames is to use the two least–significant bits of the pad octet that accompanies the IV field. KeyID=00 would specify the standard key (default key or key from the key mapping array). KeyID values 01, 10, and 11 would specify alternate key values appropriate to the key management mechanism in use. The value for the KeyID field in outgoing frames would be obtained from a WEP_Key_ID MIB variable. The Default_WEP_Key would become a 4–key vector, indexed by KeyID. Stations that implement the WEP key mapping option only have to support KeyID value 00 on a per–address basis. More advanced key management mechanisms that needed alternate KeyIDs in conjunction with key mapping would only be usable within service sets where all of the stations supported the required number of KeyIDs in the key mapping array.

RECOMMENDATION: Change the definition of the pad octet in the 4–octet IV field to an octet which contains a 2–bit KeyID in the least–significant two bits and has the other 6 bits reserved. Extend the Default_WEP_Key to a 4–entry vector, indexed by KeyID.