

Security Risks of 802.11

Waleed Abdelwahab

Computer Systems Department, Faculty of Computer and Information Sciences,

Ain Shams University

wabdelwah@criticalsites.com

ABSTRACT

The 802.11 standard for wireless networks includes a Wired Equivalent Privacy (WEP) protocol to protect link layer communications from eavesdropping and other attacks. However, in its current form, WEP suffers from serious security flaws which arise due to the reuse of the Initialization Vector (IV) and the deployment of an unkeyed checksum for message authentication.

1. INTRODUCTION

The rapid spread of the laptop computers and PDA's in the recent years has caused an increase in the range of places people perform computing. At the same time, network connectivity is becoming an increasingly integral part of computing environments. As a result wireless networks of various kinds have gained much popularity.

The 802.11 standard [1] for wireless LAN communications introduced the Wired Equivalent Privacy (WEP) protocol in an attempt to address these new problems and bring the security level of wireless systems closer to that of wired ones. The primary goal of WEP is to protect the confidentiality of user data from eavesdropping.

WEP is part of an international standard; it has been integrated by manufacturers into their 802.11 hardware and is currently in widespread use. Unfortunately, WEP falls short of accomplishing its security goals. Despite employing the well-known and believed-secure RC4 [2] cipher, WEP contains several major security flaws. The flaws give rise to a number of attacks, both passive and active, that allow eavesdropping on, and tampering with, wireless transmissions. In this paper, we discuss the

flaws that we identified and describe the attacks that ensue.

2. Wired Equivalent Privacy (WEP) protocol

The wired equivalent privacy protocol is used in 802.11 networks to protect link-level data during wireless transmission. It's described in details in 802.11 standards [1]; and the following is a brief description to enable the following discussion of its properties.

WEP relies on a secret key K shared between the communicating parties to protect the body of a transmitted frame of data. Encryption of a frame proceeds as follows

Checksum: First, we compute an integrity checksum $c(M)$ on the message M We concatenate the two to obtain a plaintext $P = (M, c(M))$, which will be used as input to the second stage. Note that $c(M)$, and thus P , doesn't depend on the key K .

Encryption: in the second stage, we encrypt the plaintext P derived above using RC4, we choose an initialization vector (IV) v and the key K . The RC4 algorithm generates a keystream - i.e., a long sequence of pseudorandom bytes - as a function of the IV v and the key K . The Keystream is denoted by $RC4(v,k)$. Then, we exclusive-or (XOR, denoted by \oplus) the plaintext with the key stream to obtain the ciphertext:

$$C = P \oplus RC4(v,k)$$

Transmission: Finally, we transmit the IV and the ciphertext over the radio link.

The Format of the encrypted frame is also shown

pictorially in Figure 1.

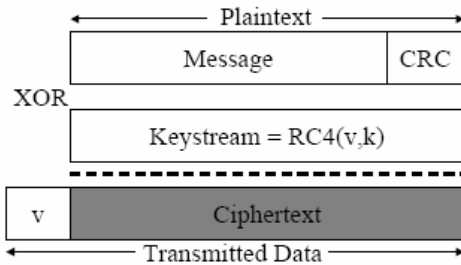


Figure 1: Encrypted WEP frame

To decrypt a frame protected by WEP, the recipient simply reverses the encryption process. First, he regenerates the keystream $RC4(v,k)$ and XORs it against the ciphertext to recover the initial plaintext:

$$\begin{aligned}
 P' &= C \oplus RC4(v,k) \\
 &= (P \oplus RC4(v,k)) \oplus RC4(v,k) \\
 &= P
 \end{aligned}$$

Next, the recipient verifies the checksum on the decrypted plaintext P' by splitting it into the form (M', c') , re-computing the check-sum $c(M')$, and checking that it matches the received checksum c' . This ensures that only frames with a valid checksum will be accepted by the receiver.

2.1 WEP Security Goals

The WEP Protocol is intended to enforce three main security goals [1]:

Confidentiality: The fundamental goal of WEP is to prevent casual eavesdropping.

Access control: A second goal of the protocol is to protect access to a wireless network infrastructure. The 802.11 standard includes an optional feature to discard all packets that are not properly encrypted using WEP, and manufacturers advertise the ability of WEP to provide access control.

Data integrity: A related goal is to prevent tampering with transmitted messages; the integrity checksum field is included for this purpose. In all three cases, the claimed security of the protocol

“relies on the difficulty of discovering the secret key through a brute-force attack” [1].

There are actually two classes of WEP implementation: classic WEP, as documented in the standard and an extended version developed by some vendors to provide larger keys. The WEP standard specifies the use of 40-bit keys. This key length is short enough to make brute-force attacks practical to individuals and organizations with fairly modest computing resources. However, it is straightforward to extend the protocol to use larger keys, and several equipment manufacturers offer a so-called “128-bit” version (which actually uses 104-bit keys, despite its misleading name). This extension renders brute-force attacks impossible for even the most resourceful of adversaries given today’s technology. Nonetheless, we will demonstrate that there are shortcut attacks on the system that do not require a brute-force attack on the key, and thus even the 128-bit versions of WEP are not secure.

2.2 Practicality of attacks

Before describing the attacks, we would like to discuss the feasibility of mounting them in practice. In addition to the cryptographic considerations discussed in the sections to follow, a common barrier to attacks on communication subsystems is access to the transmitted data. Despite being transmitted over open radio waves, 802.11 traffic requires significant infrastructure to intercept.

An attacker needs equipment capable of monitoring 2.4GHz frequencies and understanding the physical layer of the 802.11 protocol; for active attacks, it is also necessary to transmit at the same frequencies. A significant development cost for equipment manufacturers lies in creating technologies that can reliably perform this task.

As such, there might be temptation to dismiss attacks requiring link-layer access as impractical; for instance, this was once established practice among the cellular industry. However, such a position is dangerous. First, it does not safeguard against highly resourceful attackers who have the ability to incur significant time and equipment costs to gain access to data. This limitation is especially

dangerous when securing a company's internal wireless network, since corporate espionage can be a highly profitable business. Second, the necessary hardware to monitor and inject 802.11 traffic is readily available to consumers in the form of wireless Ethernet interfaces. All that is needed is to subvert it to monitor and transmit encrypted traffic. We were successfully able to carry out passive attacks using off-the-shelf equipment by modifying driver settings. Active attacks appear to be more difficult, but not beyond reach.

Therefore, we believe that it would be prudent to assume that motivated attackers will have full access to the link layer for passive and even active attacks. Further supporting our position are the WEP documents themselves. They state: "Eavesdropping is a familiar problem to users of other types of wireless technology" [1, p.61].

We will not discuss the difficulties of link layer access further, and focus on cryptographic properties of the attacks.

3. Keystream reuse risks

WEP provides data confidentiality using a stream cipher called RC4. Stream ciphers operate by expanding a secret key (or, as in the case of WEP, a public IV and a secret key) into an arbitrarily long "keystream" of pseudorandom bits. Encryption is performed by XORing the generated keystream with the plaintext. Decryption consists of generating the identical keystream based on the IV and secret key and XORing it with the ciphertext.

A well-known pitfall of stream ciphers is that encrypting two messages under the same IV and key can reveal information about both messages:

$$\text{If } C1 = P1 \oplus RC4(v,k)$$

$$\text{And } C2 = P2 \oplus RC4(v,k)$$

Then

$$\begin{aligned} C1 \oplus C2 &= (P1 \oplus RC4(v,k)) \oplus (P2 \oplus RC4(v,k)) \\ &= P1 \oplus P2 \end{aligned}$$

In other words, XORing the two ciphertexts (C1 and C2) together causes the keystream to cancel out, and the result is the XOR of the two plaintexts ($P1 \oplus P2$).

Thus, keystream reuse can lead to a number of attacks: as a special case, if the plaintext of one of the messages is known, the plaintext of the other is immediately obtainable. More generally, real-world plaintexts often have enough redundancy that one can recover both P1 and P2 given only $P1 \oplus P2$; there are known techniques, for example, for solving such plaintext XORs by looking for two English texts that XOR to the given value $P1 \oplus P2$ [3]. Moreover, if we have n ciphertexts that all reuse the same keystream, we have what is known as a problem of *depth n*. Reading traffic in depth becomes easier as n increases, since the pairwise XOR of every pair of plaintexts can be computed, and many classical techniques are known for solving such problems (e.g., frequency analysis, dragging cribs, and so on) [4,5].

Note that there are two conditions required for this class of attacks to succeed:

- A The availability of ciphertexts where some portion of the keystream is used more than once, and
- A Partial knowledge of some of the plaintexts.

To prevent these attacks, WEP uses a per-packet IV to vary the keystream generation process for each frame of data transmitted.

WEP generates the keystream $RC4(v,k)$ as a function of both the secret key k (which is the same for all packets) and a public initialization vector v (which varies for each packet); this way, each packet receives a different keystream. The IV is included in the unencrypted portion of the transmission so that the receiver can know what IV to use when deriving the keystream for decryption. The IV is therefore available to attackers as well, but the secret key remains unknown and maintains the security of the keystream.

The use of a per-packet IV was intended to prevent keystream reuse attacks. Nonetheless, WEP does not achieve this goal. We describe below several realistic keystream reuse attacks on WEP. First, we discuss how to find instances of keystream reuse; then, we show how to exploit these instances by taking advantage of partial information on how typical plaintexts are expected to be distributed.

3.1 Decryption Dictionaries

Once the plaintext for an intercepted message is obtained, either through analysis of colliding IV's, or through other means, the attacker also learns the value of the keystream used to encrypt the message. It is possible to use this keystream to decrypt any other message that uses the same IV. Over time, the attacker can build a table of the keystreams corresponding to each IV. The full table has modest space requirements—perhaps 1500 bytes for each of the 2^{24} possible IV's, or roughly 24 GB—so it is conceivable that a dedicated attacker can, after some amount of effort, cumulate enough data to build a full decryption dictionary, especially when one considers the low frequency with which keys are changed. The advantage to the attacker is that, once such a table is available, it becomes possible to immediately decrypt each subsequent ciphertext with very little work.

Of course, the amount of work necessary to build such a dictionary restricts this attack to only the most persistent attackers who are willing to invest time and resources into defeating WEP security. It can be argued that WEP is not designed to protect from such attackers, since a 40-bit key can be discovered through brute-force in a relatively short amount of time with moderate resources. However, manufacturers have already begun to extend WEP to support larger keys, and the dictionary attack is effective regardless of key size. (The size of the dictionary depends not on the size of the key, but only on the size of the IV, which is fixed by the standard at 24 bits.) Further, the dictionary attack can be made more practical. Since typical use of wireless cards includes re-initialization at least once per day, building a dictionary for only the first few thousand IV's will enable an attacker to decrypt most of the traffic directed towards the access point. In an installation with many 802.11 clients, collisions in the first few thousand IV's will be plentiful.

3.2 Key Management

The 802.11 standard does not specify how distribution of keys is to be accomplished. It relies on an external mechanism to populate a globally-

shared array of 4 keys. Each message contains a key identifier field specifying the index in the array of the key being used. The standard also allows for an array that associates a unique key with each mobile station; however, this option is not widely supported. In practice, most installations use a single key for an entire network.

This practice seriously impacts the security of the system, since a secret that is shared among many users cannot stay very well hidden. Some network administrators try to ameliorate this problem by not revealing the secret key to end users, but rather configuring their machines with the key themselves. This, however, yields only a marginal improvement, since the keys are still stored on the users' computers. As anecdotal evidence, we know of a group of graduate students who reverse-engineered the network key merely for the convenience of being able to use unsupported operating systems.

The reuse of a single key by many users also helps make the attacks in this section more practical, since it increases chances of IV collision. The chance of random collisions increases proportionally to the number of users; even worse, wireless cards that reset the IV to 0 each time they are reinitialized will all reuse keystreams corresponding to a small range of low-numbered IV's. Also, the fact that many users share the same key means that it is difficult to replace compromised key material. Since changing a key requires every single user to reconfigure their wireless network drivers, such updates will be infrequent. In practice, we expect that it may be months, or even longer, between key changes, allowing an attacker more time to analyze the traffic and look for instances of keystream reuse.

4. Message Authentication

The WEP protocol uses an integrity checksum field to ensure that packets do not get modified in transit. The checksum is implemented as a CRC-32 checksum, which is part of the encrypted payload of the packet.

We will argue below that a CRC checksum is insufficient to ensure that an attacker cannot tamper with a message: it is not a cryptographically

secure authentication code. CRC's are designed to detect random errors in the message; however, they are not resilient against malicious attacks. As we will demonstrate, this vulnerability of CRC is exacerbated by the fact that the message payload is encrypted using a stream cipher.

4.1 Message Modification

First, we show that messages may be modified in transit without detection, in violation of the security goals. We use the following property of the WEP checksum:

PROPERTY 1: The WEP checksum is a linear function of the message.

By this, we mean that check summing distributes over the XOR operation, i.e., $C(x \oplus y) = C(x) \oplus C(y)$ for all choices of x and y this is a general property of all CRC's checksum.

One consequence of the above property is that it becomes possible to make controlled modifications to a ciphertext without disrupting the checksum. Let's fix our attention on a ciphertext C which we have intercepted before it could reach its destination:

$$A \rightarrow B : (v, c)$$

We assume that C corresponds to some unknown message M , so that

$$C = RC4(v, k) \oplus (M, c(M)) \quad (1)$$

We claim that it is possible to find a new ciphertext C' that decrypts to M' , where $M' = M \oplus \Delta$ and Δ may be chosen arbitrarily by the attacker. Then, we will be able to replace the original transmission with our new ciphertext by spoofing the source,

$$(A) \rightarrow B : (v, C')$$

And upon decryption, the recipient B will obtain the modified message M' with the correct checksum.

All that remains is to describe how to obtain C' from

C so that C' decrypts to M' instead of M . The key observation is to note that stream ciphers, such as RC4, are also linear, so we can reload many terms. We suggest the following trick: XOR the quantity $(\Delta, c(\Delta))$ against both sides of equation 1 to get new ciphertext C' :

$$\begin{aligned} C' &= C \oplus (\Delta, c(\Delta)) \\ &= RC4(v, k) \oplus (M, c(M)) \oplus (\Delta, c(\Delta)) \\ &= RC4(v, k) \oplus (M \oplus \Delta, c(M) \oplus c(\Delta)) \\ &= RC4(v, k) \oplus (M', c(M \oplus \Delta)) \\ &= RC4(v, k) \oplus (M', c(M')) \end{aligned}$$

In this derivation, we used the fact that the WEP checksum is linear so that

$$c(M) \oplus c(\Delta) = c(M \oplus \Delta)$$

as a result we have shown how to modify C to obtain a new ciphertext C' that will decrypt to $P \oplus \Delta$

This implies that we can make arbitrary modifications to an encrypted message without fear of detection. Thus, the WEP checksum fails to protect data integrity, one of the three main goals of the WEP protocol (see Section 2.1).

Notice that this attack can be applied without full knowledge of M : the attacker only needs to know the original ciphertext C and the desired plaintext difference Δ , in order to calculate $C' = C \oplus (\Delta, c(\Delta))$. For example, to flip the first bit of a message, the attacker can set $\Delta = 1000 \dots 0$. This allows an attacker to modify a packet with only partial knowledge of its content.

4.2 Message Injection

Next, we show that WEP does not provide secure access control. We use the following property of the WEP checksum:

Property 2: The WEP checksum is an unkeyed function of the message.

As a consequence, the checksum field can also be computed by the adversary who knows the message. This property of the WEP integrity

checksum allows the circumvention of access control measures. If an attacker can get ahold of an entire plaintext corresponding to some transmitted frame, he will then be able to inject arbitrary traffic into the network. As we saw in Section 3, knowledge of both the plaintext and ciphertext reveals the keystream. This keystream can subsequently be reused to create a new packet, using the same IV. That is, if the attacker ever learns the complete plaintext P of any given ciphertext packet C , he can recover the keystream used to encrypt the packet:

$$P \oplus C = P \oplus (P \oplus RC4(v,k)) = RC4(v,k)$$

He can now construct an encryption of a message M' :

$$(A) \rightarrow B : (v, C')$$

Where

$$C' = (M', c(M')) \oplus RC4(v,k)$$

Note that the rogue message uses the same IV value as the original one. However, we can appeal to the following behavior of WEP access points:

Property 3: It is possible to reuse old IV values without triggering any alarms at the receiver.

Therefore, it is not necessary to block the reception of the original message. Once we know an IV v along with its corresponding keystream sequence $RC4(v,k)$, this property allows us to reuse the keystream indefinitely and circumvent the WEP access control mechanism.

A natural defense against this attack would be to disallow the reuse of IV's in multiple packets, and require that all receivers enforce this prohibition. However, the 802.11 standard does not do this.

While the 802.11 standard strongly recommends against IV reuse, it does not require it to change with every packet. Hence, every receiver must accept repeated IV's or risk non-interoperability with compliant devices. We consider this a flaw in the 802.11 standard.

In networking one often hears the rule of thumb "be conservative in what you send, and liberal in what you accept." However, when security is a

goal, this guideline can be very dangerous: being liberal in what one accepts means that each low-security option offered by the standard must be supported by everyone, and is thus available to the attacker. This situation is analogous to the ciphersuite rollback attacks on SSL [6], which also made use of a standard that included both high-security and low-security options. Consequently, to avoid security at the least-common denominator level, we suggest that the 802.11 standard should be more specific about forbidding IV reuse and other dangerous behavior.

Note that in this attack we do not rely on Property 1 of the WEP checksum (linearity). In fact, substituting any unkeyed function in place of the CRC will have no effect on the viability of the attack. Only a keyed message authentication code (MAC) such as SHA1-HMAC [7] will offer sufficient strength to prevent this attack.

4.2 Authentication Spoofing

A special case of the message injection attack can be used to defeat the shared-key authentication mechanism used by WEP. The mechanism is used by access points to authenticate mobile stations before allowing them to form an association. After a mobile station requests shared-key authentication, the access point sends it a challenge, a 128-byte random string, in cleartext. The mobile station then needs to respond with the same challenge encrypted using WEP. The authentication succeeds if the decryption of the response calculated at the access point matches the challenge. The ability to generate an encrypted version of the challenge is considered proof of possession of a key.

However, as described in the previous section, it is possible to inject properly encrypted WEP messages without the key. All that is necessary is knowledge of a plaintext/ciphertext pair of the requisite length. It is easy to obtain such a pair by monitoring a legitimate authentication sequence: the attacker learns both the plaintext challenge sent by the access point and the encrypted version sent by the mobile station. From this, it is easy to derive the keystream used to encrypt the response. Since all authentication responses are of the same

length, the recovered keystream will be sufficient to create a proper response for a new challenge (received in plaintext).

Therefore, after intercepting a single authentication sequence using a particular key, the attacker can authenticate himself with that key indefinitely. This is a particularly serious problem when the same shared key is used by all mobile stations, which is frequently the case in practice.

5. Countermeasures

There are configuration options available to a network administrator that can reduce the viability of the attacks we described. The best alternative is to place the wireless network outside of the organization firewall. Instead of trying to secure the wireless infrastructure, it is simpler to consider it to be as much of a threat as other hosts on the Internet. The typical clients of a wireless network are portable computers that are mobile by their nature, and will frequently employ a Virtual Private Network (VPN) solution to access hosts inside the firewall when accessing via dial-up or from a remote site. Requiring that the same VPN be used to access the internal network when connected over 802.11 obviates the need for link-layer security, and reuses a well-studied mechanism. To provide access control, the network can be configured such that no routes to the outside Internet exist from the wireless network. This prevents people within radio range of the wireless infrastructure from usurping potentially costly Internet connection bandwidth, requiring VPN use for any outside access. (However, it may be desirable to allow visitors to access the Internet wirelessly without additional administrative setup.)

A useful additional measure is to improve the key management of a wireless installation. If possible, every host should have its own encryption key, and keys should be changed with high frequency.

The design of a secure and easy-to-use mechanism for automated key distribution to all users is a good subject for further research. Note, though, that good key management alone cannot solve all of the problems described in this paper; in particular, the attacks from section 4 remain applicable.

6. lessons

The attacks in this report serve to demonstrate a fact that has been well-known in the cryptography community: design of secure protocols is difficult, and fraught with many complications. It requires special expertise beyond that acquired in engineering network protocols. A good understanding of cryptographic primitives and their properties is critical. From a purely engineering perspective, the use of CRC-32 and RC4 can be justified by their speed and ease of implementation. However, many of the attacks we have described rely on the properties of stream ciphers and CRC's, and would be rendered ineffective, or at least more difficult, by the use of other algorithms. There are also more subtle interactions of engineering decisions that are not directly related to the use of cryptography. For example, being stateless and being liberal in what a protocol accepts are well-established principles in network engineering. But from a security standpoint, both of these principles are dangerous, since they give an attacker more freedom to operate, and indeed, the traffic injection attacks capitalize on this freedom. Security is a property of an entire system, and every decision must be examined with security in mind.

The setting of WEP makes a secure design particularly difficult. A link-layer protocol must take into account interactions with many different entities at the same time. The IP redirection attack relies on collaboration between an agent injecting messages at the link layer and a host somewhere the Internet. The complex functionality of a 802.11 access point makes it susceptible to such attacks from all sides. Faced with such difficulties, even the most experienced of security professionals can make serious errors. Recognizing this fact, the accepted practice is to rely on the expertise of others to improve the security of protocols. Two important ways to do this is to reuse past design and to offer new designs for public reviews.

Past designs should be reused whenever possible. A common tenet of protocol design is "don't do it." WEP could have benefited from the experience gained in the design of the IP Security protocol (IPSEC) [8]. Although the goals of IPSEC are

somewhat different, it also aims to provide link-layer security, and as such needs to deal with many of the same issues as WEP. Even if the Protocol could not be reused as-is, a review of its design and past analysis would have been very instructive. Some of the previously published problems in IPSEC [9] share many similarities with the attacks presented in this paper.

Public review is also of great importance. If WEP had been examined by the cryptographic community before it was enacted into an international standard, many of the flaws would have been almost surely eliminated. (For example, the dangers of using a CRC to ensure message integrity are well-known [10].) While we applaud the fact that the standard is open, there are still barriers to public review. A security researcher is faced with a financial burden to even attempt to examine the standard—the cost of the document is in the hundreds of dollars. This is the opposite of what should be—a working group developing a new security protocol should proactively invite the security community to analyze it.

7. Conclusions

In this report, we have demonstrated major security flaws in the WEP protocol and described several practical attacks that result. Consequently, we recommend that WEP should not be counted on to provide strong link-level security, and that additional precautions be taken to protect network traffic. We hope that our discoveries will motivate a redesign of the WEP protocol to address the vulnerabilities that we found. Our further hope is that this paper will expose important security principles and design practices to a wide audience, and that the lessons we identify will benefit future designers of both WEP and other mobile communications security protocols.

8. Acknowledgments

I'd like to thank Prof. Dr. Ahmed Hamad for helping and supporting with many articles and papers.

9. REFERENCES

- [1] 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.
- [2] R. L. Rivest. The RC4 Encryption Algorithm. RSA Data Security, Inc., Mar. 12, 1992. (Proprietary).
- [3] E. Dawson and L. Nielsen. Automated cryptanalysis of XOR plaintext strings. *Cryptologia*, (2):165–181, Apr. 1996.
- [4] S. Singh. The code book: the evolution of secrecy from Mary, Queen of Scots, to quantum cryptography. Doubleday, New York, NY, USA, 1999.
- [5] W. Tutte. FISH and I, 1998. A transcript of Tutte's June 19, 1998 lecture at the University of Waterloo.
- [6] D. Wagner and B. Schneier. Analysis of the SSL 3.0 protocol. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce (EC-96)*, pages 29–40, Berkeley, Nov. 18–21 1996. USENIX Association.
- [7] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-hashing for message authentication. RFC 2104, Feb. 1997.
- [8] S. Kent and R. Atkinson. Security architecture for the Internet Protocol. Internet Request for Comment RFC 2401, Internet Engineering Task Force, Nov. 1998.
- [9] S. M. Bellovin. Problem areas for the IP security protocols. In *6th USENIX Security Symposium*, San Jose, California, July 1996. USENIX.

[10] Core SDI. crc32 compensation attack against ssh-1.5.

<http://www.coresdi.com/soft/ssh/attack.txt>, July 1998.

[11] W. A. Arbaugh. An inductive chosen plaintext attack against WEP/WEP2. IEEE Document 802.11-01/230, May 2001.

[12] Jin-Cherng Lin. Secure enhanced wireless transfer protocol, April 2006

[13] "WIRELESS SECURITY IN 802. 11 (WI-FI®) NETWORKS" WHITE PAPER, January 2003 by Dell

[14] N. Borisov, I. Goldberg and D. Wagner, Intercepting mobile communications: the insecurity of 802.11, Proc. of the 7th Annual International Conference on Mobile Computing and Networking, July 16-21, 2001.

[15] P Nicopolitidis, M S Obaidat, G I Papadimitriou and A S Pomportsis, Wireless networks, Wiley, 2003.