WEP Vulnerabilities and Attacks

Alex Blank
adb3160@cs.rit.edu
4005-706 Cryptography II
5 May 2010

Abstract. The Wired Equivalent Privacy algorithm, sometimes incorrectly referred to as the Wireless Encryption Protocol, is a method of securing IEEE 802.11 wireless internet connections. It was introduced in 1997 and became the standard for wireless security. Over the next several years, however, the WEP protocol and its underlying cryptographic primitives were scrutinized and consequently revealed to be vulnerable on a number of levels. Numerous attacks were formulated to exploit these weaknesses and demonstrate the insecurity of WEP. It was superseded by the current standard, WPA, in 2004; nevertheless, the transition away from usage of WEP has been slow, leaving many networks open to easy intrusion. This paper will describe the vulnerabilities of WEP as well as give an account of several available attacks.

1. Introduction

Wireless networks have been in use since World War II in the form of radio communication systems, and have progressed steadily alongside the inception of various technologies requiring data transfer in a mobile environment, such as GSM. With the birth of the Internet came a new and prominent usage for wireless networks, allowing users to connect to LAN infrastructure through access points without wiring. Initially, the security of data transfer over these connections was an afterthought. As the Internet grew in popularity and wireless networks gained greater usage, however, the need for a solution to secure transmission became apparent. In 1997, this need gave rise to the Wired Equivalent Privacy, or WEP, algorithm, which became the IEEE standard for wireless encryption.

The WEP protocol's reign as wireless security standard lasted only a few years. This was the result of close scrutiny of the WEP protocol by the scientific community, and in only a few years time a critical vulnerability was found with WEP's underlying cryptographic primitive, the stream cipher RC4. Further analysis over the following years showed not only could RC4 be used to compromise the protocol, but the protocol itself was vulnerable to the structure of 802.11.

The structure of this paper is as follows: Section 2 contains an overview of the WEP algorithm, while Section 3 delves into further detail about its vulnerabilities. Section 4 identifies an outlines several attacks that have been developed to exploit WEP. Section 5 explores the current day relevance of continued usage of WEP, and finally, Section 6 offers some conclusions.

2. The WEP Algorithm

WEP relies on a secret key k that is shared between computers connected to the network and the network access point. Originally k was relegated to 64 bits due to governmental constraints, but was later expanded once those constraints were lifted. The key k is formed of two parts, the *root key* of length 40 bits, and a randomly chosen *initialization vector* (IV) of length 24 bits [1]. k is then fed to the RC4 stream cipher and used to initialize its internal state. RC4's PRGA uses the internal state to generate a string of bits called the *keystream*. The plaintext, composed of the data and a 4 byte CRC32 checksum, is then XORed with the keystream to create an encrypted

payload [1]. Finally, the IV is concatenated in plaintext to the front payload and the packet is sent over the network. Figure 1 gives a high level overview of the WEP encryption process.

Decryption requires retrieving the IV from the received packet, concatenating it with root key, generating the keystream with RC4 and XORing the ciphertext portion of the received packet with the keystream to give the plaintext.

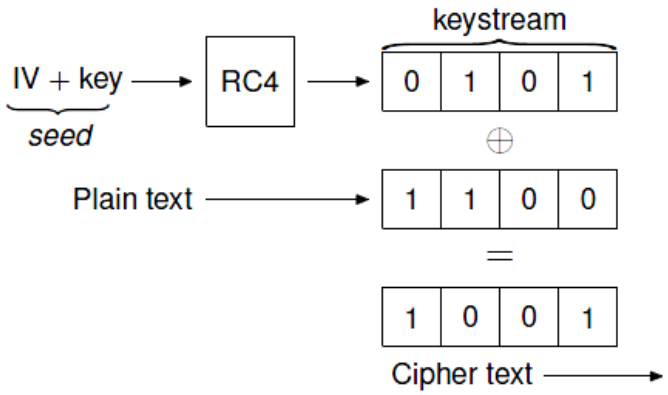

Figure 1

3. Vulnerabilities

WEP has several vulnerabilities. Fluhrer, Mantin and Shamir showed in 2001 that the RC4 algorithm used to generate the keystream for encryption is subject to two separate weaknesses: a) the existence of a large class of weak keys for which a few bits of secret key *k* can potentially reveal a substantial amount of the initial permutation of the internal state, and b) a related key vulnerability (which is more the focus of our attention) stemming from the observation that, when used in a mode similar to WEP where *k* is split into an IV and a root key, if the root key is used with multiple different IVs, an attacker can compute the root key by analyzing the initial word of the corresponding keystreams [6]. Another rather apparent vulnerability was found to exist due to the probability of IV reuse, caused by the smaller size of the IV [1]. In 2006, Bittau, Handley and Lackey demonstrated a different type of vulnerability in WEP, this one appearing in the structure of 802.11 in the form of packet fragmentation and the predictable format of ARP response packets [4]. Vulnerabilities also exist in that the main cryptographic weaknesses of WEP are not related to the actual root key itself, and therefore improvements to the protocol involving key lengthening

are insufficient. Finally, the CRC32 checksum was found to cause a partial break due to the linear nature of the RC4 cipher and its relationship to computations in polynomial rings [3]. The following section and its subsections detail attacks that can be used to exploit some of these vulnerabilities.

4. Attacks

Since WEP was discovered to be a vulnerable algorithm, many different attacks have been invented to exploit and demonstrate its weaknesses. These have gained progressive speed as years have gone by, with the Fluhrer, Mantin and Shamir (FMS) attack requiring on the order of millions of packets to the Tews, Weinmann and Pyshkin (TWP) attack which requires only tens of thousands of packets and offers ARP injection to stimulate network traffic [6, 2]. This paper details only three of the many attacks discovered, but includes a diversity of approaches in order to give a clearer picture of the many different types of holes poked in WEP.

4.1 FMS Attack

Regarding the related key vulnerability, Fluhrer, Mantin and Shamir devised a method of using the weakness of RC4, the keystream generator for WEP, to obtain bits of the root key based on probabilities of certain state permutation bits reaching a "resolved state" such that they contained a correlation to the first byte of the keystream. First, the stream cipher RC4 will be described, followed by an overview of the attack itself.

4.1.1 RC4

RC4 is one of the most widely used stream ciphers today. It was invented by Ronald Rivest in 1987 and leaked to the public seven years later. It utilizes an internal state S, represented as an array, which in practice has a length of 256 bytes. RC4 is composed of two parts: a Key Scheduling Algorithm (KSA) and a Pseudorandom Generation Algorithm (PRGA). The user inputs a key (in WEP, this is the IV concatenated with the root key) into the KSA, which then initializes S to the identity S[i] = I and then generates a “random” permutation of S using the key. The PRGA then operates a loop that

swaps values of S and then chooses a byte of the S in a pseudorandom fashion. The algorithm is illustrated in figure 2.

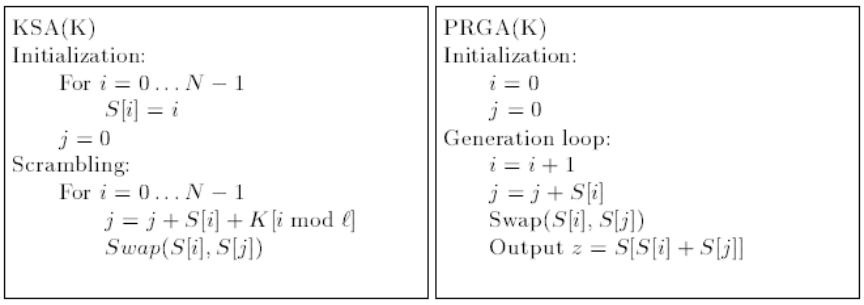

Figure 2

4.1.2 Related Key Attack

There are two attacks outlined in [6]; we will focus on the one stressing the related key weakness. This attack takes advantage of the public nature of the IV in WEP and assumes that at least the first word of the keystream can be obtained.

In that we are initially dealing with only the first word of the keystream Z, it can be shown to be generated by S[S[1] + S[S[1]]]. This is shown in figure 3

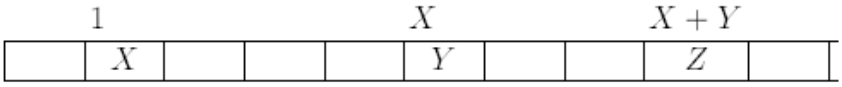

Figure 3

where

$X = S_i[1]$
$Y = S_i[S_i[1]]$ and
$Z = S_i[S_i[1] + S_i[S_i[1]]]$

A *resolved* condition is then defined as satisfying the following:

$i \geq 1$
$X = S_i[1]$
$X + Y = S_i[1] + S_i[S_i[1]]$

Fluhrer et al calculate that X + Y (= Z) will be output as the first word of the output of RC4 with probability 0.05, or 5% in this situation. The idea of the attack is to observe IVs that fit the resolved condition

and use the first word correlation to gain information about the root key.

Once a packet has been intercepted, the attacker first determines whether the current IV, of length I, is in a resolved condition. This can be done easily by simply computing the first I rounds of RC4 KSA and checking against the resolved condition criteria. If the IV is in a resolved condition, then the B^{th} bit of the root key K can be generated from the following equation:

$$K[B] = S^{-1}{}_{1+B-1}[Out] - j_{I+B-1} - S_{I+B-1}[I + B]$$

This can be refined for the first byte of K (B = 1):

$$K[B] = S^{-1}{}_{1}[Out] - j_{I} - S_{I}[I + 1] \quad (1)$$

where S^{-1} denotes an inverse permutation which yields, in this case, the location of Out in the permutation S_1.

The attacker uses the IV to compute S up to the I^{th} round. If the attacker knew the value at $S_{I+1}[I+1]$, he would know its location in S_I, which is the value of j_{I+1}, and would then be able to compute K[I + 1], which is the first bit of the root key. The attacker does not know S_{I+1}, but does know that the IV is in the resolved condition and that the first bit of the root key can be recovered in this state via (1) with probability 0.05. By collecting enough IVs, an attacker can eventually derive the first root key byte, and then iteratively derive subsequent bytes across the entire root key.

This attack was one of the first formulated against WEP, albeit indirectly through RC4. It requires on the order of millions of packets to collect enough IVs for a sufficiently high probability of recovering K[B]. Tews and Beck point out that a major flaw in this attack is its restriction to IVs containing the resolved condition, as IVs that do not meet those requirements are discarded [5]. Later attacks improved on this model of attack and significantly decreased the number of packets needed to recover the entire root key.

4.2 Fragmentation Attack

The Fragmentation attack proposed by Bittau, Handley and Lackey utilizes the 802.11 framework against WEP to both transmit and decrypt data on a secured network [4]. The concept is less cryptographically inclined, but is yet another example of varied nature of the vulnerabilities present in protocol.

The attack begins by intercepting a packet on the network, preferably an ARP packet. With regards to a general packet, it can be said that the first 8 bytes of plaintext can be known due to a common LLC/SNAP header on each encrypted packet. Since the first 8 bytes of the plaintext is known, it can be XORed with the ciphertext to give 8 bytes of the keystream. For the case of an ARP packet, more is known about its format and thus more keystream bits may be extracted. The keystream can now be used to encrypt data, although the use of this is infeasible when considering that an 8 byte allocation is sectioned into 4 bytes of payload and 4 bytes of CRC32 checksum. Thus, the attack must utilize the 802.11 feature of packet fragmentation to send a protocol maximum of 16 four byte fragments, thus generating a 64 byte packet for injection. Figure 4 illustrates this method.

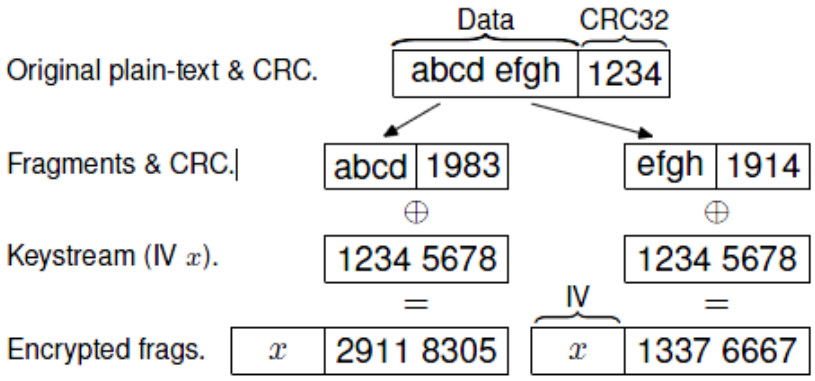

Figure 4

Decryption of packets on the network requires a bit more setup, but is possible with a connection to the Internet and a controlled host in the Internet domain. After collecting an encrypted packet, the attacker computes the 8 bytes of keystream. The keystream is used to generate a fragmented IP header (figure 5) with a destination address pointed at the Internet host controlled by the attacker; this header along with the payload portion of the original encrypted packet is sent along to the access point, which decrypts, defragments and sends along to the destination. Because WEP only applies for traffic on the wireless network, Internet traffic is not encrypted and thus the attacker is privy to the unencrypted payload.

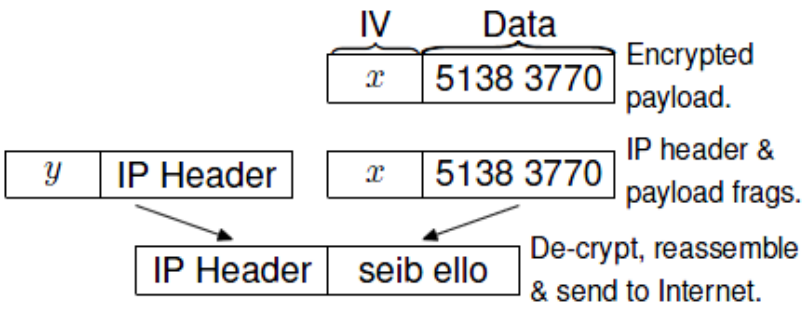

Figure 5

This attack is notable because it does not rely on knowledge of the root key to act as a functioning host on the network, with the ability to send and receive data.

4.3 Chopchop Attack

In a 2004 posting on netstumbler.org, a person going by the pseudonym of KoreK posted an attack, described in greater detail in [3], which utilizes the mathematical structure of the CRC32 checksum that is appended to the end of the plaintext payload before encryption. The premise of this attack is as follows: assuming the last 4 bytes of the encrypted packet comprise the CRC32 checksum, remove the last byte and replace it with a "corrected" guess that replicates a valid checksum, then send it off to the access point. If the access point returns with an error message, the guess was incorrect and another is made. If the access point returns with confirmation, then the final byte has been guessed correctly and is now known to the attacker. This can be extended to the other bits of the checksum as well, providing a partial break.

First, a look at the math. An arbitrary bit string can be represented as coefficients of a polynomial in GF(2). A given polynomial P is said to have a correct checksum if and only if the following equation holds:

$$P \bmod R_{CRC} = P_{ONE} \quad (1)$$

where R_{CRC} is the polynomial representing the CRC checksum and P_{ONE} represent a 32 bit string of all ones. Note that P_{CRC} is irreducible in GF(2). We can then write P as the sum of the first three bytes, denoted Q, shifted 8 bits left added to the last byte, denoted P_7.

$$P = QX^8 + P_7 \quad (2)$$

From this we can separate the first three bytes from the last byte and treat the first three bytes as a full checksum by adding a byte shift. This gives Q an invalid checksum value, and so it must be corrected. Q will be correct when the following condition is met:

$$Q = P_{ONE} \bmod R_{CRC} \quad (3)$$

or

$$Q \bmod R_{CRC} = P_{ONE} \quad (4)$$

Plugging (1) into (2) gives

$$P_{ONE} \bmod R_{CRC} = QX^8 + P_7$$

or

$$QX^8 = (P_{ONE} + P_7) \bmod R_{CRC} \quad (5)$$

We can then define an inverse to X^8, since it can be inverted in the polynomial ring, as $(X^8)^{-1} = R_{INV}$. Applying RINV to both sides of (5), we get

$$Q = R_{INV}(P_{ONE} + P_7) \bmod R_{CRC} \quad (6)$$

Pointing back to (4), we can then deduce that the addition of P_{COR} to Q will correct the checksum, where

$$P_{COR} = P_{ONE} + R_{INV}(P_{ONE} + P_7)$$

because

$$Q = (R_{INV}(P_{ONE} + P_7) + P_{ONE} + R_{INV}(P_{ONE} + P_7)) \bmod R_{CRC}$$

$$= P_{ONE} \bmod R_{CRC}$$

Which fulfills the condition in (3) and thus the checksum is correct. Since the correction value is dependent on P_7 we do not know its value, we must makes guesses, of which there are at most 256 per byte. On average an attacker will only need 128 guesses, and thus only an average of 128 packets is required. Guessing can continue for the last *m* bytes of the packet with an average of 128*m* needed for decryption.

5. Current Relevance

While WPA has been the standard for wireless network encryption for the past 6 years, WEP is still very much in use. A survey of wireless networks in London and Seattle yielded a WEP / WPA usage of 76% / 20% and 85% / 14%, respectively [4]. Another survey

conducted in 2007 showed that 46.3% of networks detected in a German city used WEP [2]. My own survey of networks in my surrounding residential area showed 54% of all detectable wireless networks to be encrypted using WEP (Figure 6).

WEP	WPA	Unsecured
6	4	1
54%	36%	9%

Figure 6

The results of this small survey are obviously not to be seen as a direct correlation to current wireless network encryption methods nation or worldwide. However, taken in perspective with the other more comprehensive surveys, it is clear that even 6 years after its full deprecation and removal as an industry standard, WEP continues to be utilized to an unusual degree considering its known insecurity. This insecurity was demonstrated in the real world when, in 2007, a wireless network encrypted with WEP and owned by retailer TJ Maxx was compromised and led to the theft of vast amounts of credit card information [7]. It is imperative that industry professionals and individuals setting up their own private networks understand the security implications of WEP and the many vulnerabilities that it presents to attackers.

6. Conclusions

Wireless networks have existed for decades, providing a method of mobile data transfer. These connections, however, were unsecured until the first wireless security standard was adopted as WEP. Unfortunately, under closer scrutiny WEP proved to be insecure due to a number of vulnerabilities presented through several different facets of the protocol. The vulnerabilities include a propensity for weak keys and a weakness related to the public nature of the IV portion of secret key fed to RC4. IV reuse was also identified as an issue due to its computationally small space ($2^{24} \approx 16M$). Issues were also found within the surrounding protocol of 802.11 and its use of predictable ARP and IP packet headers which easily give away bytes of the keystream. These vulnerabilities led to numerous attacks, including those created by Fluhrer, Mantin and Shamir which yields the whole root key, a fragmentation attack presented by Bittau, Handley and Lackey which allows encryption and decryption of packets without knowledge of the root key, and the Chopchop attack

discovered by Korek which presents a partial attack through exploitation of the CRC32 checksum. These attacks reinforce the deprecation of this algorithm in 2004, yet WEP is still in wide use today. It is clear from the variability of its vulnerabilities and the wide range of attacks available that WEP must not be used anymore by consumers looking to secure their wireless networks. Industry standards have changed and it is imperative that current wireless networks reflect that change.

References

[1] Borisov, Goldberg, Wagner. Intercepting Mobile Communications: The Insecurity of 802.11, 2001.

[2] Tews, Weinmann, Pyshkin. Breaking 104 bit WEP in less than 60 seconds, 2007.

[3] Tews. Attacks on the WEP protocol, 2007.

[4] Bittau, Handley, Lackey. The Final Nail in WEP's Coffin, 2006.

[5] Tews, Beck. Practical Attacks Against WEP and WPA, 2008.

[6] Fluhrer, Mantin, Shamir. Weaknesses in the Key Scheduling Algorithm of RC4, 2001.

[7] Max. TJ MAXX Update : Weak Wi-Fi Encryption (WEP) To Blame For The Security Breach, http://www.bestsecuritytips.com/news+article.storyid+226.htm, 2007.