Enhanced WEP: a new solution to WEP threats

Hani Ragab Hassan, Yacine Challal

Enhanced WEP: A new efficient solution to WEP threats

Hani Ragab Hassan and Yacine Challal

***Abstract*—With the rapidly increasing importance of wireless networks, there have been many recent proposals, dealing with the insecurity of the Wired Equivalent Privacy (*WEP*) protocol. In our paper, we have analyzed threats' origins, and proposed a new solution based upon the WEP, which achieves, in addition to security goals originally aimed at by WEP, another security service which is replay detection. Our premise is to permit deploying an efficient security mechanism on wireless networks, without reconsidering all the security architecture. Contrary to the standards WPA proposed by the Wi-Fi Alliance and IEEE 802.11i proposed by the Task Group I of IEEE 802.11, our solution requires neither hardware add-on nor replacement, but merely software updates.**

***Index Terms*—Wireless networks, security, protocol analysis, WEP.**

I. Introduction

WITH the widespread use of wireless networks, securing data transmission becomes a basic requirement. The IEEE 802.11 standard which defines wireless networks communication, has proposed in its second version IEEE 802.11*b* a new protocol to offer some wired-like security services, such as: data privacy, data integrity, and authentication. Unfortunately, this protocol falls short these objectives, and has shown many threats which were exploited by intruders. The Task Group I started developing a more secured standard: the IEEE 802.11*i*. Meanwhile, the Wi-Fi alliance Group together with IEEE proposed the Wi-Fi Protected Access (WPA), which enhances security model of WEP using the well known authentication . Despite their efficiency, these two standards, and especially 802.11*i*, need hardware renew and reconsideration of security architecture.

This paper begins with an introduction of WEP's well-known vulnerability, followed by a description of our solution, and a comparaison between the two.

II. WEP

The WEP was designed by a group of IEEE volunteer members, aiming at giving some layer of security to wireless networks, this layer offers following services:

1. Data Privacy: it's the basic service offered by the WEP, transiting data can be read by only communication authenticated members;
2. Data Integrity: WEP offers a guarantee to the receiver that data wasn't altered;
3. Authentication: depends strongly on data integrity, an corrupted message is considered as non authenticated and is automatically rejected.

Compiegne University of Technology
Computer Science Department
Heudiasyc lab. France
Phone: +33 (0)3 44 23 44 23
emails: {hragabha,ychallal}@hds.utc.fr

A. WEP's Security Mechanisms

In this section, we'll describe WEP functioning process, which includes mechanisms used to implement different security services. Initially, both of the communication entities share a secret key k. k will be somehow used farther to encrypt transmitted data. Let A be a source who tries to send a message M to a receiver B. A begins by calculating a checksum using the CRC algorithm widely used in network protocols. Let's note T=(M,CRC) the message produced by a simple concatenation of M and its CRC.

Par la suite, A encrypts T using the RC4 algorithm [1 ou 2 ref]. RC4 proceeds by making stream ciphers. It generates a keystream R using two inputs:

- The key k shared between A and B, it's made of 40 bits;
- An *Initialisation Vector iv*, used principally to minimize probability of feeding RC4 with the same entries (which leads to the same keystream in output).

R is XORed with T to produce the cipher text C. To decrypt C, B needs to reconstruct the same keystream R and XOR it with C, indeed:

$$C \bigoplus R = (T \bigoplus R) \bigoplus R = T \bigoplus (R \bigoplus R) = T$$

However, to reproduce R, B needs to know *iv*. The WEP has expected to concatenate *iv* to C. Encrypting process is shown in fig. Note that *iv* is sent as clair text, without any kind of encryption. This process ensures:

- Data Privacy: all transmitted data are encrypted, only communication entities can decrypt them;
- Data Integrity and Authentication: the checksum is verified when the message arrived. Thus, all modifications of the message in its path will be detected.

B. Attacks

All the security model of WEP is based upon its resistance against brute-force attacks [6]. There're currently two implemented variants of the WEP [5]:

- Classical WEP: as defined by IEEE, the key length is 40 bits;
- 128 bits WEP: it's a version proposed by manufacturers, where the key length is 104 bits, the remaining 24 bits are reserved for *iv*.

It's important to note that the 128 bits version, and even its name, offers only a 104 bits security. In fact, the part devoted to *iv* is transmitted as clair text on the wireless network. Some vendors have already made this mistake

with the classical WEP while claiming that their products offer a 64 bits security.

The basic threat in the WEP is due to a property of stream ciphering. Indeed, in XOR based stream cipher, there's a golden role to respect: *"Reuse of keystream is forbidden"* . It's for this purpose that iv field was added, because the key k change rarely. The problem lies in the following property: let R be a keystream obtained with k and some initialisation vector *iv*, T_1 and T_2 two messages we want to transmit, C_1 and C_2 encrypted messages corresponding to T_1 and T_2 respectively. So, we can show that $C_1 \bigoplus C_2 = M_1 \bigoplus M_2$.

Indeed, $C_1 \bigoplus C_2 = (M_1 \bigoplus T) \bigoplus (M_2 \bigoplus T)$
$= (M_1 \bigoplus M_2) \bigoplus (T \bigoplus T)$
$= M_1 \bigoplus M_2$.

This property means that if a malicious person knows C_1 and C_2 (they can be easily obtained by eavesdropping) and M_1, he can guest C_2 by simple XORs. However, he would find some difficulties before being able to decrypt exchanged messages. One question becomes imperative: "is it possible to avoid reuse of initialisation vectors?" In theory, the answer is no. It's obvious that the number of values of *iv* is limited. An access point emitting packets of 1500 B with a rate of 5 MB/s (about 45% of the maximal bandwidth), will inevitably reuse some *iv* in less that 12 hours.[Intercepting Mobile]

In practice, reusing an *iv* is frequent. Some manufacturers of Wi-Fi cards reset *iv* to zero every time that the card starts. Thus, big values of *iv* will be rarely used, while small values will be used more often than not. Other manufacturers generate *iv* randomly, which seems to be a good solution for reset problem. Nonetheless, this solution will probably reuse the same *iv* in less than only 5000 transmitted packets (problem known as *birthday paradox*), this solution should be discarded.

The attacker should also detects reuse of initialisation vectors. This is very simple to do, *iv*s are transmitted as clair text. Now that the attacker has two or more encrypted messages corresponding to some *iv*, he has to find one original message to be able to decrypt the others using only XOR operations

There're many ways to obtain un clair message. The simplest one is to send some packets (requests, emails, ...) to the target, and eavesdrop the network to detect possible *iv* reusing.

Another mean is to analyze IP packets. In fact, many fields of these packets are known, and take more or less commun values. The attacker can also try to guess information contained in some messages, for example prompts used for authentication (login, passwords, ...)[5].

A determined attacker can construct a complete dictionary for decrypting. If we suppose that for every possible value of *iv* (there're in all 2^24 value), he will store the corresponding keystream (that he has already obtained using one of the above methods), a keystream is of 1500 Bytes. Hence, the dictionary size will be 24 GB [5].

We should mention that the attacker doesn't know at all k, but rather all keystreams that can be obtained from this k, using all possible values of *iv*. Thus we deduce that using a 104 bits key is pointless against this attack.

C. Threats Sources Analysis

Once that the RC4 cipher is decrypted, none of the security services can be guaranteed. The CRC isn't signed. Thus, the attacker can decrypt, then modify or even forge his own messages, then recompute corresponding CRC, and impersonalize some communication entity. Thereby making data integrity and authentication services obsoletes.

Our analysis has provided as with the fact that all WEP weakness comes from four principal conception flaws:

i. The initialization vector is transmitted as clair text.

Beside the fact that this weakens the power of encrypting, attackers are in a position to detect every *iv* reuse.

ii. The key is rarely renewed.

Key (k) updating techniques are completely leaved as implementation detail. Thus, manufacturers are free to use technique that they find adapted. The worst, an implementation that doesn't plan key renewing is within the norm.

iii. The CRC isn't signed.

As mentioned above, using non signed CRCs allows attackers to forge their own messages. Using Message Authentication Code (MAC) would be an efficient solution to this problem. Another solution is to secure enough the privacy mechanism, so that nobody will be able to access the CRC.

iv. Security services are all implemented using only one mechanism.

All the security scheme is based upon the strength of the mechanism of data privacy service. Thus, once that the privacy of data is broken, all other services - data integrity and access control- are directly broken.

In the following section we will propose a solution to bypass WEP flaws.

III. Hidden Initialisation Vector WEP: *HIVW*

HIVW aims to resolve WEP flows problem. And this, without changing hardware used, and keeping a good interoperability with existing WEP. In fact, in addition to the scheme that we propose to allow establishment of security enhanced channels, between two nodes that uses HIVW, we propose another scheme to ensure interoperability in the case where only one node uses HIVW. In this section we will describe how HIVW achieves these two goals.

A. Encryption Process

We propose a scheme that looks like the WEP's one. The difference is that in HIVW we encrypt both of the message T and *iv* with RC4. So we use a keystream which is longer with 24 bits.

First, the two nodes agree on some initial *iv*, this step is detailed in section B. After this, the sender S generates randomly a new *iv*. Let iv_1 be the generated *iv*, and iv_0 the initial *iv*. S uses his own key k and iv_0 to generate a keystream KS using RC4.

S concatenates the CRC to the message M which gives T=(M,CRC), then he concatenates iv_1 to T. The all is then XORed with KS. S sends the XORed message to the receiver R.

Knowing iv_0, R decrypts the message, verifies CRC, and then stores iv_1. iv_1 will be used to decrypt the next packet sent by S. As a matter of fact, every packet contains the initialisation vector used to encrypt the next one. So, the packet p_i encrypted using iv_{i-1} contains iv_i, which will be used to encryption and decryption of p_{i+1}.

Initialisation Vectors are all, except the first one, generated randomly by the sender. This makes no correlation between ivs, and therefore enhances resistance to brute-force attacks.

B. Initialisation of iv

The process shown above requires that S and R agree on the initial iv. This can be done by many ways. We suggest here two methods that are enough secure and don't produce an overhead.

The first manner is to agree on the iv using Diffie-Hellman (DH) algorithm [1 ou 2 rf DH]. This algorithm allows two entities to establish a private key, based upon messages exchanged publicly. This algorithm proceeds as follow... . It will be sufficient to take as iv_0 the first 24 bits of the key agreed using DH.

The second manner is to use an hash algorithm [1 ou 2 ref] to compute iv. A hash algorithm is an one-way function that allows to compress some bitstream BS into a digest D of a given length. The principle is to compute D in each side hashing a common information, such as k. Thus, S and R compute separately the digest of k, and take the first 24 bits as iv_0. *SHA-1* which is a secured well-known hash algorithm can be use to this end.

C. Key Renewing and Distribution

The WEP isn't provided with an efficient rekeying mechanism. We have proposed rekeying algorithm that renew and distribute the key according to one (or both) of two types of constraints. The first one is the exchanged quantity of data. For example, we can decide to change k every 1 GB. The second is time passed from last rekey. For example, every 4 hours. It's obvious that such mechanisms will harden attackers task.

Rekey is done using special messages. These messages have a flag of 8 bytes set to zero followed bye the new key, and the kind of constraints to use for the next renew. The new key can be generated randomly to minimize generation overhead.

D. Improving CRC (change title)

To ensure data integrity and access control even if data privacy mechanism is broken, we propose a scheme that separates the two mechanism (data privacy, data integrity and access control). We suggest that S and R will have an integrity key k_i used together with the initialisation vector iv to generate another keystream KS_i using RC4. This keystream will be used to encrypt M before computing its CRC. Thus, even the data privacy scheme is broken. It's still hard to intruders to forge or even modify transiting packets. This process is shown in figure [fig.]

E. Reliability

We mean here by reliability, the answer to the question *"How to do if some packet is lost?"*. You have already noted surely that if some HIVW traffic packet is lost, all packets coming from the same sender wouldn't be decryptable. This is due to the fact that every packet contains a part of the key used to decrypt its successor. Packets constitute a sort of a chain.

When a packet is lost, the sender S should resend it. So, we can design another special message M_l, which is sent by a receiver when he detects that a message was lost. M_l contains the sequential number of lost packet. So, we impose to the two parts of a communication to store the sequential number of the last sent or received message. The detection of message lost is simply done when R receives an illegible message. So he supposes that there was a problem somewhere, and he sends a message M_l containing the sequential number of last received message SQN_R. When S receives M_l, he restarts emission from SQN_R. That implies that S has to buffer last sent messages. Given that we're working at link level, the number of buffered messages will not harm to memory stockage of S

F. Security Services

The scheme that we have shown above ensures following services, which are services originally aimed at by the WEP, and a supply service which is replay detection.

F.1 Data Integrity

Data integrity is ensured using the CRC of encrypted M. Even if the data privacy service is broken, our protocol still ensures data integrity, provided that we modify a little bit our scheme. To explain why, let's suppose that a clever intruder has broken the data privacy mechanism. Now he has probably the iv used to encrypt the packet, and the message M that it contains, and the CRC of M$\oplus KS_i$. So, he can obtain some information about M$\oplus KS_i$ using the CRC, that he can use somehow later to break data integrity scheme. To avoid running any risk, rather that including M in every packet, M$\oplus KS_i$ is sent, concatenated with its CRC, and the iv used to encrypt the next packet. In this way, the global encryption scheme is modified. Note that if the optional encryption of M before computing CRC is chosen, this modification doesn't imply any overhead, because the computation of M$\oplus KS_i$ is done already.

F.2 Data Privacy

Now that we have seen how the data integrity is ensured, we remark that there is a double layer of encryption of every message sent (the second one stills optional). Breaking the first layer is almost impossible. Because key is regularly changed, that doesn't let enough time to attackers to construct a dictionary, especially as they have no idea about the iv used. Even though the second layer

isn't necessary, it makes attackers task harder. We suggest to add it only when the option of using enhanced CRCs is checked.

F.3 Access Control

As in WEP, access control in HIVW depends strongly on efficiency of data integrity service. So, every packet that isn't integre, as considered non authenticated is rejected.

F.4 Replay Detection

Replay detection is simply implemented by verifying whether the received packet is decryptable or not. If the packet is a replay, it can't be decrypted by the current *iv* because it changes for every packet.

IV. Interoperability with existing WEP

V. Fitting it all together

In HIVW, we use keys that has a total length of 128 bits when added to *iv*s. And as *iv* isn't transmitted as clear text, we can change *iv*s length to 64 bits or even 128 bits to make our protocol more dynamic. The reason of *iv*s length limitation in original WEP was the fact that the *iv* was transmitted as clear text, so sending a longer key will considerably weaken the global scheme.

VI. Comparaison with WEP

In this section, we compare WEP with HIVW according to three criteria. The first is level of security guaranteed, the next one is the bandwidth overhead, and finally the computation overhead.

A. Security

Security scheme of WEP is already broken, and as we've shown in F.2, privacy mechanism of HIVW is very resistant against intrusions. The fact that HIVW separates security mechanisms makes it more robust.

B. Bandwidth Overhead

Given that key updating messages are sent rarely comparing to data messages, and that HIVW doesn't add any supplementary field to packets exchanged, we can affirm that bandwidth overhead of HIVW is the same as WEP.

C. Computation Overhead

Before speaking about computation overhead, we should explain the reason of using keystreams in the WEP. In fact, using keystreams allows to separate computation in two different steps. The first one is done off-line, and is the generation of the keystream. The second step is the XORing the keystream to the message. Using a keystream has the advantage to reduce computation to do between receiving a message and sending it. RC4 is particulary fast and simple, it can be seen as a simple random numbers generator.

In HIVW we respect this constraint, so only keystreams are used. Computation overhead is reduced to the cost of computing additional 24 bits in KS which can be neglected given that RC4 doesn't require a considerable computation. And computation of the optional KS_i which still be very cheap because it uses RC4. The user has the choice between enhancing security, or reducing a little bit computation overhead.

VII. Conclusion

VIII. Not access control but message authentication