On Enhancing the WEP Security Against Brute-force and Compromised Keys

Saif Ur Rehman, Saeed Ullah, and Sardar Ali
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST), Islamabad, Pakistan
Email: {saifur.rehman, saeed.ullah, sardar.ali}@seecs.edu.pk

***Abstract*—The IEEE 802.11 standard uses Wired Equivalent Privacy (WEP) for data encryption in wireless Local Area Networks. So far, different flaws have been discovered in the security of WEP. Frequently changing the encryption key can improve the security of WEP but there is no built-in provision for this in the standard. In this paper first we present and critically review different possible methods of automatic key updating and then propose a dynamic key management technique. The proposing technique works at the application layer. It is an automated encryption key updation method that can significantly improve the security of WEP without requiring any changes in the standard or at the lower layers of the OSI model.**

***Keywords*-WEP, security, dynamic key management, wireless networks security.**

I. Introduction

Wireless communications are becoming ubiquitous in homes, offices, educational institutions and public places like airports, and train stations etc; with its ability to provide high-speed, high-quality information exchange between portable devices. The IEEE 802.11 is one of the most popular Wireless Local Area Network (WLAN) widely used worldwide. In WLAN, users perform data sharing without the need to connect to a common wireline network. Since wireless transmission is broadcasting by nature, therefore data security can be easily compromised. Attackers, with a device capable of understanding the protocol, could observe the data flow and compromise the privacy of the network.

For secure authentication and data protection, the IEEE 802.11 standard includes an encryption system called Wired Equivalent Privacy (WEP) [7] - [10]. There are three main participants in WLAN: Supplicant, Authenticator and Authentication Server. Supplicant is the one looking to get access to the wireless network while Authenticator is the access point which realizes the access control and allows only the authenticated Supplicants to use the network. Authentication Server is a centralized server which authenticates the supplicants. In this mechanism the network participants (stations) and the Access Point (AP) relies on a common set of shared keys. Due to an inherent design flaw and general misuse of the WEP options, 802.11 data security has been successfully broken with relatively little effort [1] - [5].

After the insecurities of WEP were discovered, different alternative encryption methods were developed and implemented in the standard. However, the fact remains that 802.11 and hence WEP is still widely used in wireless ethernet networks. Networks which are using this standard still use WEP for encryption and are therefore exposed to the security loopholes that are inherent to WEP [10]. Apart from these weaknesses, wireless being a broadcast domain is also prone to man-in-the-middle attack [1].

WEP encryption is optional in the standard. To improve the security of the standard, WEP encryption should be implemented. The rare key updation is another problem that weakens the WEP security. Updating a key requires an administrator to inform all users who wish to use the network. This method of key distribution increases the risk of exposing the key to intruders and attackers.

In this paper we present different automatic key updating techniques that would allow for frequent key updates. In the end we propose a key generation technique which can provide a large number of encryption keys at user's end. This technique eliminates the need of key transfer over the wireless domain. We mathematically analyze that the key space of our proposing technique is large enough and is hence unfeasible to be compromise through brute force attack.

The remainder of the paper is organized as follows: Section II covers background and related work. In section III, major automatic key updating techniques are discussed along with their pros and cons. Section IV introduces our proposed methodology, and section V formulates the generalized mathematical expression of the proposed solution. The paper is concluded in section VI.

II. Background and related work

WEP uses a stream cipher RC4 for encryption. Block diagram of WEP encryption is shown in figure 1. RC4 has a key of length 128 bits, entered as 26 hexadecimal characters, each of which represents 4 bits of the key. This sum up to 104 bit, the remaining 24 bits are used for the initialization vector to change the key stream generated for that key. Some vendors are also providing support for 256 bit keys. However the 128 bit key length is the most widely deployed. Key length is an important factor of any encryption algorithm; longer key lengths exponentially increase the required brute force attack efforts. The combined protection against brute

978-1-4244-7818-7/10/$26.00 ©2010 IEEE

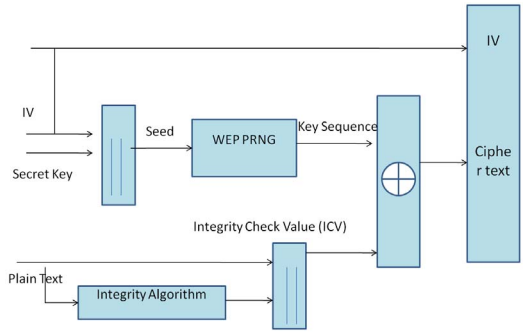

Figure 1. WEP block diagram

force, provided by RC4 using 128 bits is

$$2^{(104+24)} = 2^{128} \quad (1)$$

$$\approx 3.4028 * 10^{38} \quad (2)$$

While this is a large enough number, unfortunately there are many weaknesses in the key scheduling algorithm of RC4. In [2] Fluhrer et. al. has shown that a large number of weak keys and some smaller part of key being more significant than a larger part. In [1] Nikita et. al. have discussed the security loopholes in 802.11 WEP such as Initialization Vector (IV), keystream reuse, dictionary attacks, possible message modification, message injection, and message redirection. Jin et. al. proposed some improvements for the security of WEP in [3], but the paper does not address the aspect of automatic key updating. AT&T labs have published a technical report [6] on successfully employing the attack described in [2].

Most wireless networks change their key very rarely once it has been implemented. The attackers can easily crack the key through brute force. Apart from this weakness caused by the size of encryption key, the Initialization vector is just 24 bits long, which is exhausted after 16777216 packets been sent.

An encryption algorithm is safe if, brute force attack on the key is infeasible either financially or temporally. Therefore a simple way of enhancing the security of WEP is to update the key frequently and automatically, thereby making brute force attacks infeasible temporally. This would also discourage the malicious nodes from trying a dictionary attack on the keystream as RC4 creates the cipher text by XOR of the keystream with the plain text. The plain text can be recovered by the XOR operation of the keystream with the cipher text. Whereas XOR of the Plain Text with the cipher text yields the keystream.

The problem with frequent updating of encryption key is that the 802.11 standard does not have any provision for that; In this paper we have solved it by referring the task to an application level program designed for the purpose.

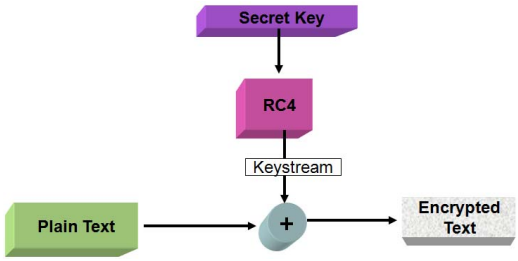

Figure 2. Encryption mechanism of RC4

III. Different Key management techniques

In order to frequently change the encryption key, we can use some of the simple and obvious methods. In the following subsections we have discussed them in detail.

A. List of encryption keys

A pre configured list of keys to be used at both client and server, based on time of the day, such as a key that changes every hour or a different key for every day of the week, or any other criteria that might be deemed fit. This would enable a node to know which key to use if it comes up after being inactive for a period in which the encryption key would have changed. For protection purpose, it is recommended to keep the keys encrypted so that the keys cannot be compromised even if a malicious node can gain access to a legitimate node.

A drawback of this system is that once the list of keys is exhausted either the list should be reused or the new list of keys has to be fed into the nodes. New list can either be transmitted securely using the existing key, or be feeded manually in every node. In secure transmission if the existing key had been compromised, that will provide the list of the future keys to the malicious node. The manual list updating for a significantly large network can be unfeasible due to the magnitude of the task.

B. On the fly key updating

Instead of providing a list of keys, the server can update/change the keys as and when one is used for a predetermined duration of time. The server would maintain a log for the encryption keys used against the time slot in which they were used. When an inactive node approaches for association, it would also send the time at which it was active last. The server would then send a challenged packet encrypted with the key k_i for that time slot (k_i was the key used by the node when it was active last time). the requesting node would decrypt the challenged packet and should perform some specific operation such as a simple permutation for example and send it back encrypted with the same key k_i. The server would then send the current key k_n encrypted with the key k_i. This key updation process is shown in figure 3. This technique has the weakness that an

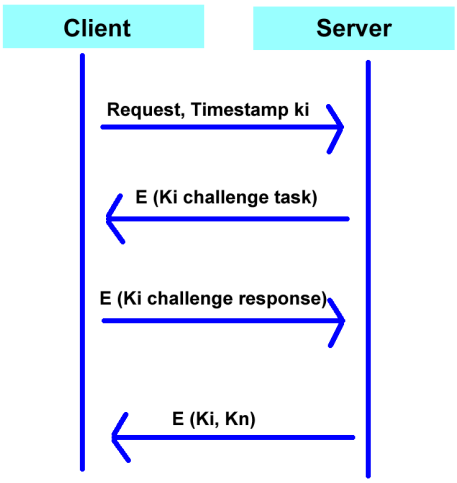

Figure 3. Sequence of messages between client and server in on the fly key updation

unauthentic node can crack a key using brute force and then authenticate using that key along with the time stamp of that key's use.

C. User specific keys

Different users can be assigned different encryption keys, however that would add extra maintenance and overburden the server.

D. MAC Authentication

Along with these methods, MAC authentication can be used, to deter MAC spoofing, any node that requests association but cannot validate in a predetermined number of attempts should be put in a wait period.

E. Public key cryptography

Public key cryptography is used for session establishment. Since the domain is broadcasting in wireless networks, a man in the middle attack renders it useless for a first time connection establishment. If there is some information shared between the server and client, authentication can be done using that information and the key be exchanged using public key cryptography.

IV. The Proposing Secure Automatic Key Updation

Keeping the drawbacks of the above mentioned solutions in view. We propose a solution which provides a simple mechanism to eliminate brute force attacks by frequently changing the encryption keys automatically. This automatic key updation is based on pre determined intervals or conditions such as wrapping around of initialization vector. It is resilient against compromised keys by generating unused, new encryption keys. The mechanism requires a one time setup, after which the legitimate node can update the encryption key automatically, even after a substantially long downtime.

Both the server and the legitimate nodes have a key updation application which comprises of a key generator and source file of length n bits. Upon receiving a request for establishing a secure encrypted connection, the server responds with a packet containing m pointers. Each pointer serves three purposes

1) It informs the requesting node of the number of segments in which the encryption key is to be generated which is determined by the number of pointers m.
2) It Informs the requesting node about the starting point of each segment in the source file.
3) It informs the requesting node about the "direction" of a segment.

A legitimate node having the source file and the key generation application then generates the encryption key for itself by selecting each segment according to its pointer and concatenating them together in sequence.

This algorithm of generating keys can generate a very large number of keys using a very small source file. The mathematical expression for calculating the number of possible keys is derived in the following section.

V. Rationale

WEP uses RC4 stream cipher for encryption. RC4 has a key length of 128 bits, let us assume a source of 1000 random bits in length. From this source file keys for RC4 will be generated using wraparound to the start of file upon reaching its end. That is, after the 1000^{th} bit, the 1st bit is selected and then the 2^{nd} and so on. This means, that each bit in the file can act as a starting point for a key. Therefore the number of keys that can be generated from the source is equal to its length.

$$n = 1000$$

$$k = 1000$$

where n is source length and k is keys generated.

Now, if the direction of bit selection is reversed, that is from 998^{th} bit is selected after 999^{th} which is selected after 1000^{th} and so forth, another n keys can be generated from the same file.

So keys generated for both direction are shown by

$$k = 1000 + 1000$$

Now let us assume that the key is generated in two halves. Both halves having independent starting points in the source.

Segments generated for the first half

$$i = 1000$$

Segments generated for the second half

$$j = 1000$$

Total possible keys for two segments

$$\begin{aligned} k_2 &= (i1|j1) + (i1|j2) + \cdots + (i1|j1000) + (i2|j1) + \cdots \\ &\quad +((i2|j1000) + \cdots + (i1000|j1000) \\ &= 1000 * 1000 \\ &= n * n \\ &= n^2 \end{aligned} \tag{3}$$

When this process is repeated for three segments

$$k_3 = n^3$$

Therefore

$$k_m = n^m \tag{4}$$

Where m represent the number of segments.

Now if two directions are considered for equation 3, we get the following possible combination.

$$\begin{aligned} k_{bd} &= \rightarrow 1000* \rightarrow 1000 \\ &\quad + \rightarrow 1000* \leftarrow 1000 \\ &\quad + \leftarrow 1000* \rightarrow 1000 \\ &\quad + \leftarrow 1000* \leftarrow 1000 \end{aligned}$$

where The $\rightarrow$ and $\leftarrow$ show the direction of bit selection.

Due to two possible options in direction, we notice that there is a similarity to the binary numbers. When we had one segment, it gave two different possibilities for k. Now with two segments the number has increased to 4.

The relation between binary number and its instances or bits is shown by equation 5

$$2^m \tag{5}$$

Where m is the number of segments.

Combining equations 4 and 5 we get

$$k_{tot} = k_m * 2^m$$

Putting value of k_m we get

$$k_{tot} = n^m * 2^m \tag{6}$$

Equation 5 shows that the number of possible keys that can be generated from a source file is exponentially proportional to the values of n and m For example the proposed solution can produce 17592186044416 different keys from a file of just 1024 bits using just 4 segments. This number of keys is large enough to provide new keys for more than 30 million years, even if a key is changed every minute. The upper bound for this solution is of course the maximum possible keys for RC4 which is given, for 128 bit encryption key as

$$2^{128} \approx 3.4 * 10^{38}$$

It can be argued that some keys might be regenerated or repeated due to real random selection rather than pseudo random selection. However, the overhead of pseudo random selection is easily avoidable. To show the feasibility of random key generation we consider the probability of re-generation of a key. This probability of generating a particular key $P_{(kg)}$ is given by

$$P_{(kg)} = 1/K$$

This means the probability of key re-generation $P_{(kr)}$ on a random scale is given by

$$P_{(kr)} = P_{(kg)} * P_{(kg)} \tag{7}$$

Putting value of $P_{(kg)}$ into equation 7 from above example we get

$$P_{(kr)} = (1/K) * (1/K)$$

As show, the probability of regeneration of a key is extremely low, therefore we can use random key generation to avoid storage and processing overhead involved in pseudo random key generation. The server just need to inform the requesting node of the number of segments, direction and initiation point of each segment. All this information can be accommodated in a few bytes of data, without causing any significant over head in the network traffic.

An attacker can mount a dictionary attack on the source file of encryption keys. Assuming that the attacker can capture all traffic within the network. The attacker would launch a brute force attack on an encryption key, the pointers for which it would store. Once the key is cracked, the attacker has obtained 128 bits of the source file. By continuing this approach, and careful selection of keys to break the attacker can eventually build the source file. The attacker would select to attack keys with minimum overlapping bits, amongst each segment and with those of already cracked keys. This can be determined by the values and directions provided in the pointers. In an ideal case, the attacker would get to know a maximum of 128 bits of the source file after successfully cracking a key. A 1024 bit source file can be regenerated by the attacker in a minimum of 8 successful cracked keys.

To mitigate this possible attack, the length of the source file can be kept significantly large. If the source file is $1MB = 8 * 1024 * 1024$ bits $= 8388608$ bits, it would take a minimum of 65536 successful keys cracked to regenerate the source file. This is a large number or successful attacks required and an unrealistic target for any attacker.

VI. Conclusion

In this paper we presented and mathematically analyzed the automatic key updation method which has the ability of frequently updating the encryption keys securely. This key updation method works totally on application layer and do not require any change to the underline standard or any lower layer of the OSI model. The proposed method generates considerable number of encryption keys from a single source file of a few kilobytes that can be used for a very long time. The proposed method have the ability of frequently updating

the encryption keys securely, which improves the security of WEP against dictionary attacks considerably in 802.11 WLAN.

REFERENCES

[1] Nikita Borisov, Ian Goldberg, and David Wagner, "Intercepting Mobile Communications: The Insecurity of 802.11", In Proceedings of ACM Mobicom 2001.

[2] Scott Fluhrer, Itsik Mantin, and A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4", LNCS, vol. 2259, pp. 124. Springer, Heidelberg (2001)

[3] Jin-Cherng Lin, Yu-Hsin Kao, and Chen-Wei Yang, "Secure enhance wireless transfer protocol", In Proceedings of the First International Conference on Availability, Reliability and Security, 2006.

[4] Matsuura K and Imai H, "Modified aggressive mode of internet key exchange resistant against denial-of-service attacks", IEICE Transactions on Information and Systems, Volume E83-D(5), pages 972-979, May, 2000.

[5] Denial of Service vulnerability in IEEE 802.11 wireless devices, http://www.auscert.org.au/render.html?it=4091.

[6] Adam Stubblefield, John Ioannidis, and Aviel D. Rubin, "Using the Fluhrer Mantin and Shamir Attack to break WEP", Technical Report, AT&T Labs TD-4ZCPZZ, August 21, 2001.

[7] 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.

[8] IEEE P802.11i/D10.0. Medium Access Control (MAC) security enhancements, amendment 6 to IEEE standard for information technology telecommunications and information exchange between systems Local and metropolitan area networks Specific requirements Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications. April, 2004.

[9] IEEE standard 802.1x for local and metropolitan area networks port-based network access control national communications system technology and programs division (N2) PO Box 4502 Arlington, Virginia 22204-4502 March 2005

[10] IEEE Standard for Information technology Telecommunications and information exchange between systems Local and metropolitan area networks Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Fast Basic Service Set (BSS) Transition. July, 2008.

[11] Changhua He and John C Mitchell "Analysis of the 802.11i 4-way handshake", ACM WiSE, 2004.