SECURING WIRELESS NETWORKS WITH ENHANCED WEP

Ramzi A. Haraty and Walid El Horr
Lebanese American University
Division of Computer Science & Mathematics
P. O. Box 13-5053 Chouran
Beirut, Lebanon 1102 2801
Email: {rharaty, walid.horr@lau.edu.lb}

ABSTRACT

Wireless is the next generation networking technology. The security of such technology is very critical because its physical layer is the electro-magnetic waves that can be easily intercepted by anyone. The original security protocol for 802.11 wireless networks is called WEP (Wired Equivalent Privacy), it is a protocol that is based on symmetric-key encryption algorithm RC4 (Rivest Cipher 4). Unfortunately this protocol had many flaws that made it exposed to many attacks. This paper presents a set of improvements for WEP: these improvements convert WEP to a hybrid cryptosystem, a system that is based on both a symmetric-key algorithm and an asymmetric-key algorithm.

KEY WORDS

802.11, Wireless, Security, WEP, Cryptography, Public-Key, Private-Key.

1. INTRODUCTION

Back in 1999, wireless networking started its growth with the publication by the Institute of Electrical and Electronics Engineers (IEEE) of the first commercially practical wireless standards. Today, as the cost of wireless technology declined and the quality increased, it became cost-effective for users and enterprises to integrate it.

This technology offers flexibility, mobility, cost reduction as well as a fairly high transmission rate. Wireless networking does, however, have security concerns: this technology creates the possibility of accidental or intentional intrusions into a network, because of the nature of the physical layer of this technology. To overcome this problem, the IEEE 802.11 committee implemented a protocol that claims to offer comparable confidentiality to a traditional wired network, hence its name Wired Equivalent Privacy: WEP is a symmetric cipher that uses the same key to encrypt and decrypt. This key is composed of two parts: the Shared Key and a variable part called the Initialization Vector that changes for each frame.

As wireless networks began to grow in popularity, a group of scientists discovered serious flaws in the implementation of this protocol because little review was performed on it during the IEEE approval process for 802.11 security mechanisms.

In this paper we present a set of improvements for Wired Equivalent Privacy networks. The rest of the paper is organized as follows: section 2 presents the weaknesses associated with WEP. Section 3 presents modifications to RC4. Section 4 discusses the management of shared keys. Section 5 presents the authentication component. And section 6 presents the conclusion.

2. WEP WEAKNESS

WEP has many flaws, we start by explaining the process of WEP, and then we list its flaws.

2.1 WEP Process

WEP is a protocol that uses a symmetric stream cipher called Rivest Cipher 4, which is composed of the Key Scheduling Algorithm (KSA) and the Pseudo Random Generation Algorithm (PRGA). The original WEP uses a 64-bit key, but a later implementation uses a 128-bit WEP key (Rappaport, T. 2002). This key is composed of two parts: one fixed part, of length 40 or 104 bits specified by the user and called Shared Key, and another variable part, of 24 bits, called the Initialization Vector that changes for every frame and is concatenated with the Shared Key to form the entire encryption key. This implies that the same Shared Key is used to encrypt all frames, but the IV added to the Shared Key makes every key unique to make sure that no two frames are encrypted using key that was used before.

The first algorithm of WEP is the Key Scheduling Algorithm (Figure1). The goal of this algorithm is to scramble the state array S, based on the key array K.

```
1. K[] = Key array
2. Initialization of state array:
3. for i = 0 ... N - 1
4.   S[i] = i
5. j = 0
6. Scrambling:
7.       for i = 0 ... N - 1
8.         j = ( j + S[i] + K[i mod l] ) mod N
9.         Swap(S[i], S[j])
```

Figure 1 – Key Scheduling Algorithm

The second algorithm of WEP is the Pseudo-Random Generation Algorithm, or PRGA (Figure2). This algorithm uses the scrambled array S to produce keystream as long as they are needed. Each byte of the keystream is XORed with the plaintext of the frame to produce the ciphertext. When the frame reaches its end, PRGA stops generating keystream.

1. Initialization:
2. i = 0
3. j = 0
4. While need output:
5. i = (i + 1) mod N
6. j = (j + S[i]) mod N
7. Swap(S[i], S[j])
8. Output z = S[(S[i] + S[j]) mod N]
9. The byte z XORed with plaintext

Figure 2 – Pseudo-Random Generation Algorithm

For each new frame, RC4 is restarted using a new WEP key (Shared Key concatenated with an IV).

The IV is sent in clear with every frame. On the receiving station, and for decryption, it's concatenated with the Shared Key to produce the same WEP key that was used in the encryption process. This key, with RC4, would produce the same keystream, which are concatenated with the ciphertext to reproduce the plaintext.

2.2 Lack of Key Management

The 802.11 standard does not specify means of deploying Shared Keys to WLAN stations (Gast, M. 2005), other than manually configuring each one of them with the same key. On a small network, with only a few stations, this may not seem like a problem. However, for a network with ten or more WLAN devices, changing a key on each piece of equipment is a very hard task. As a result, most Shared Keys are used for an extremely long period of time, thus increasing the chances that an attacker discovers the key. To make matters worse, in most cases, these keys are never changed which results in a very weak security scheme.

2.3 IV Collision

WEP uses a three-byte IV for each frame transmitted over the WLAN. When the data is sent, the IV is pre-pended to the encrypted packet. This ensures the receiving party has all the information it needs to decrypt the data. However, if we take a closer look at the statistical nature of this process, we can quickly see a potential problem: the total size of the IV is 24 bits: if we calculate all the possible IVs, we would have a list of 2^{24} different possible keys. Although this might sound like a huge number (16,777,216), but one could expect to start seeing repeats, also known as collisions, after just about 10,000 frame transmissions.

2.4 Weak IVs

In their paper, "Weaknesses in the Key Scheduling Algorithm of RC4" (Fluhrer, S., Mantin, I. and Shamir, A. 2001), the authors have discovered a very important flaw in the use of Initialization Vector in the KSA. They indicated that the use of some IVs, called weak IVs, can leak information about the Shared Key used. This attack is based on the first output byte of the PRGA algorithm. As we know, the IV in WEP has a length of 3 bytes. The format of a weak IV is as follows:

IV byte$_1$	**IV byte$_2$**	**IV byte$_3$**
B+3	**255**	**X**

For example, the IV (3, 255, 7) is considered weak, and has the chance of leaking information on the first byte of the Shared Key (B=0).

2.5 Poor Authentication

WEP's Shared Key Authentication is based on the Shared Key: a client should prove knowledge of it in order to be authenticated. Shared Key Authentication uses a challenge send by the Access Point that the station XORs with its Shared Key to produce the challenge response. The Access Point then compares the response with its own calculations (Figure 3).

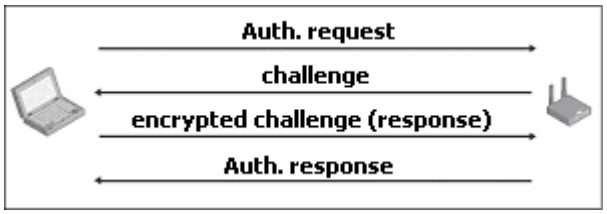

Figure 3 – WEP Authentication

In the case of the Shared Key Authentication, a hacker can sniff both the plaintext challenge and its corresponding encrypted response. Therefore, simply by XORing the two together, he has a copy of the keystream produced by RC4.

3. RC4 MODIFICATION

In its implementation in WEP, RC4 has many flaws: we force the use of 128-bit keys only, and we hash that key using MD5 to give another 128-bit output. This output key is then used in RC4.

3.1 Using 128-bit WEP Keys Only

As we saw earlier, WEP allows the use of a 40-bit or 104-bit Shared Key to be concatenated with an IV of length 24 bits to form the entire WEP key of length 64 or 128 bits which is given as input to the RC4 algorithm. In our use of RC4, we will only allow the use of the 128-bit WEP Key: an IV of length 24 bits will be concatenated

with a 104-bit Shared Key only to form the entire WEP key.

3.2 Hashing WEP Key

As we saw before, Fluhrer, Mantin and Shamir have discovered a serious flaw in WEP: some IVs called weak IVs can reveal information on the Shared Key.

Our approach is to hash the key before feeding it to KSA: MD5 (Rivest, R., 1992) will be used to take the key as input and produce a 128-bit hash value that will be the new key. This way each frames will be encrypted using a totally different key, no two frames will be encrypted using the same Shared Key part because using a different IV with the same Shared Key and hashing their concatenation would give a totally different hash result: this approach solves the problem of weak IVs because K[3]… K[15] is different for every frame (as well as K[0], K[1], K[2]).

For the sending station, the hash is easily calculated: a random IV is concatenated to the Shared Key (just like WEP works), but instead of immediately using this key with RC4, it is hashed to produce a new 128-bit key that is fed to the RC4 algorithm. For the receiving station that holds the same Shared Key, when it receives the encrypted frames, it gets the IV used. Once the IV is known, it is concatenated to the Shared Key and hashed; the result is the same hash that was produced by the sending station. This resulting hash is used as the key that is used to decrypt the encrypted frame: it produces the same keystream that was used to encrypt the frame

4. MANAGING SHARED KEYS

WEP, the original 802.11 encryption protocol does not provide any form of key management.. We present a new Key Management scheme for WEP that is based on an asymmetric cryptosystem.

4.1 Hybrid Cryptosystem

Symmetric and asymmetric algorithms each have their own advantages and disadvantages: Symmetric ciphers are much faster than asymmetric ciphers (Schneier, B. 1996), but have the problem of key distribution. Asymmetric algorithms have a pair of keys and do not worry about key distribution because the private key is never shared, but these algorithms are slow.

A Hybrid Cryptosystem is a system that combines some of the best features of both symmetric and asymmetric encryption protocols: both types of protocols are used in the same system, preserving the speed of symmetric algorithms and the security of asymmetric algorithms

Our proposed solution is to use a Hybrid Cryptosystem that keep using the RC4 symmetric encryption to encrypt every frame, but add an asymmetric cipher to act as a Key Exchange protocol, that will only be used to encrypt and distribute the changing Shared Key periodically in a secure way.

The Access Point will use the public key of each station to encrypt the RC4 Shared Key and send it to it securely. The corresponding station decrypts the Shared Key using its own private key.

4.2 RSA Cryptosystem

The RSA algorithm (Figure 4), named after its creators, Ron Rivest, Adi Shamir and Len Adleman (Rivest, R., Shamir, A. and Adleman, L. 1978), was invented in 1977 and is the most popular and widely used asymmetric encryption system today. This algorithm is a block cipher which provides security, authentication and digital signature. Its security is based on the fact that finding large prime numbers is computationally easy, but factoring the product of two such numbers appears computationally infeasible.

1. Initialize P and Q (2 large primes)
2. Calculation:
3. $N=P*Q$
4. $\varphi(N)=(P-1)(Q-1)$
5. D co-prime to $\varphi(N) \rightarrow GCD(D, \varphi(N)) = 1$
6. $E \rightarrow E*D \equiv 1 (\mathrm{mod}\ \varphi(N))$
7. Public key = (E, N)
8. Private key = (D, N)
9. Ciphertext $C = M^E \bmod N$
10. Plaintext $M = C^D \bmod N$

Figure 4 – RSA Algorithm

In comparison with ElGamal (ElGamal, T. 1985) and Rabin asymmetric cryptosystems, RSA was picked because:

1. RSA is both an encryption scheme as well as a digital signature method, unlike ElGamal that needs a separate signature system in order to digitally sign its messages.
2. Rabin's decryption is way too slow compared to the other two systems.
3. With RSA, according to the NESSIE consortium, 1024-bit RSA keys should remain "secure until at least the year 2010" (NESSIE Consortium 2003), therefore we are going to use keys of length 1024 bits.
4. The SafeXcel 1741 chipset that we'll examine in detail in the next section, supports a maximum of 1024-bit RSA key

4.3 Encryption Chipset – The SafeXcel 1741

Our hybrid system relies on the Access Point: as we will see, the Access Point will be involved in heavy RSA encryption and decryption calculations. For this reason, the much faster hardware encryption and decryption

should be used in the Access Point to match the high speed processing power of any computer today.

Many different RSA hardware chips exist in the market today. One of these is the the SafeXcel produced by SafeNet technology. The SafeXcel 1741 was chosen because it has the following features:

1. Supports MD5 hash function with a rate of 451 Mbps as well as the symmetric cipher RC4.
2. Supports the asymmetric cipher RSA:
    - 1024-bit public key encryption and signature verification in 0..85 ms
    - 1024-bit private key decryption and signature in 8.4 ms
3. Hardware based and non-deterministic Random Number Generator (RNG)
4. Can internally generate session keys (Shared Keys for our system) as well as Initialization Vectors.
5. Cost-effective: We e-mailed SafeNet and they informed us that an Access Point equipped with this chip would cost just about 25$ more than a simple WEP-only Access Point.

We assume that this RSA chip is used in the Access Points that we are going to test throughout this work.

4.4 Key Distribution

Once the asymmetric cipher was chosen to be RSA, it is time to describe how our system distributes the Shared Key to be used by stations once they are authenticated and when that key is changed by the Access Point that is equipped with the SafeXcel chipset.

4.4.1 Getting the Initial Shared Key

Upon a successful authentication, as we'll see later on, the Access Point would have all the stations' 1024-bit public keys and each station would have the Access Point's public key. The process of getting the Shared Key is:

0. At t=0, Alice checks the Shared Key in use and forms a frame containing it and with destination Bob.
1. At t=1, Alice encrypts (signs) the MD5 hash value of the frame with privateAlice to form the Digital sig. = privateAlice(MD5(frame))
2. At t=1, Alice encrypts the frame using the public key of Bob = publicBob(frame)
3. At t=2, Alice sends the encrypted frame with the signature to Bob and waits for an Acknowledgement.
4. At t=3, Bob receives the frame and decrypts it using privateBob
5. At t=4, Bob decrypts (verifies) the digital signature using publicAlice and compares the result obtained with the result of its own MD5 hash computation. If these 2 match, Bob concludes that the frame comes from Alice and is not altered.
6. At t=5, Bob sends the ACK to Alice.
7. Bob has the Shared Key now. It can start using it to encrypt frames using RC4

4.4.2 Shared Key Changing

In their paper (Stubblefield, A., Ioannidis, J. and Rubin, A. 2001), the authors recommended "securely re-keying each user after every approximately 10,000 packets": they showed that re-keying every 10,000 packets would prevent most attacks because the key is changing often.

Using a frame counter in an inactive wireless network is not efficient, therefore the better approach is to use a time counter: based on the results of different tests on the usage of 10,000 frames and on the fact that we are hashing the WEP key before using it (an additional security layer) and that the average number of stations associated to an Access Point is 30 (Masafumi O. 2003). We concluded that an interval time of 5 minutes (300 seconds) for changing the Shared Key should be enough.

Let us assume that the Access Point reached the 300-second mark and is ready to change the Shared Key on all its associated stations Bob, Clara and Dan. The steps are:

0. At t=0, Alice randomly generates a 104-bit Shared Key= SharedKey
1. At t=1, Alice encrypts every frame (different destinations) that contains the same Shared Key: (SharedKey) by the public key of every associated station: the number of different encryptions is equal to the number of associated stations.
    - publicBob(frame1)
    - publicClara(frame2)
    - publicDan(frame3)
2. At t=2, Alice encrypts the MD5 hash value of each one of these different frames with Private(Alice) to form a different Digital signature for each station.
    - privateAlice(MD5(frame1))
    - privateAlice(MD5(frame2))
    - privateAlice(MD5(frame3))
3. At t=3, Alice sends this signed encrypted Shared Key to every station (each encrypted with the corresponding public key). Now every station has 2 Shared keys: the old one in use and the newer one.
4. At t=4, every station decrypts the frame using its own private key:
    - Bob decrypts the frame with privateBob(frame1)
    - Clara decrypts the frame with privateClara(frame2)
    - Dan decrypts the frame with privateDan(frame3)

5. At t=5, every station decrypts the digital signature (signature verification) and compares the result obtained with the computed hash. If these 2 matches, every station concludes that the frame comes from Alice and is not altered.
6. At t=6, every station send an Acknowledge frame indicating a successful reception of the new Shared Key.
7. At t=7, Alice receives the ACK from every station. If an ACK frame is missing for an associated station, the Access Point retries a maximum of three times until it receives the ACK.
8. At t=8, Alice sends a frame (switchingFrame) that switches the Shared Key on every station and on the Access Point itself. The counter is restarted when the first ACK frame is received indicating a successful key switching.
9. Everyone is using the new Shared Key now

5. AUTHENTICATION

The purpose of authentication is first to prove that a station is really who it claims to be (entity authentication or simply, identification) and second, to prove message origin and integrity: makes sure that the message has not been forged and changed in the way (message authentication).

5.1 Message Authentication

Our Key Management scheme uses a public key system where two keys are generated per station and Access Point, a public key and a private key. Using this mechanism by itself does not protect against message forgery: a man-in-the-middle attack can occur when a station catches messages and send them as if it is the legitimate source. A subtle and important variant of this method can solve this problem: it requires an entity to sign its messages, therefore digital signature is needed.

Message signing works in the reverse way from encryption (Davies, J. A. & Price, W. L. 1980). The private key is used to create a signature and the corresponding public key checks the signature. In most of the cases, the hash value of a message is encrypted it with the private key. The result is added to the end of the message. Anyone who receives the message can decrypt the signature using the user's public key.

5.2 Entity Authentication

Message authentication will be used in Entity authentication and identification which makes sure that only authorized stations can join aparticular Access Point.

5.2.1 Authentication Process

The administrator logs on to the Access Point and creates a new username and his corresponding password. The password is stored as an MD5 hash. The same username and password should be used in the station to be authenticated.

To begin the authentication process:

0. At t=0, Bob probes Alice sending it publicBob.
1. At t=1 and upon receiving the public key of bob, Alice send its own public key publicAlice to Bob. If the Acknowledge packet of each key is not received, the key in question is resent. From now on all communication between these two entities is always encrypted using each other's public key.
2. At t=2, a random challenge text of the same length of the hash of the password is generated by the Access Point. (The AP has a list of all username/password pairs)
3. At t-3, Alice XORs the challenge with the hash value of the password of Bob. The result is a 128-bit length string named Result1 that is kept secret in the Access Point.
4. At t=4, Alice encrypts the frame using the public key of Bob = publicBob(frame)
5. At t=5, Alice encrypts the MD5 hash value of the frame with privateAlice to form the Digital signature = privateAlice(MD5(frame))
6. At t=6, Alice sends the encrypted frame with the signature to Bob and waits for an Acknowledgement.
7. At t=7, Bob receives the frame and decrypts it using privateBob
8. At t=8, Bob decrypts (verifies) the digital signature using publicAlice and compares the result obtained with the result of its own MD5 hash computation. If these 2 match, Bob concludes that the frame comes from Alice and is not altered.
9. At t=9, Bob sends the ACK
10. At t=10, bob XORs PassMD5(Bob) with the challenge obtained. The result is a 128-bit length string named Result2.
11. At t=11, Bob encrypts the frame using the public key of Alice = publicAlice(frame)
12. At t=12, Bob computes the MD5 hash of the frame and encrypts the result with PrivateBob to form the Digital signature = privateBob(MD5(frame))
13. At t=13, Bob sends the encrypted frame with the signature to Alice and waits for an Acknowledgement.
14. At t=14, Alice receives the frame and decrypts it using privateAlice
15. At t=15, Alice decrypts (verifies) the digital signature using publicBob and compares the result obtained with the result of its own MD5 hash computation. If these 2 match, Alice concludes that the frame comes from Bob and is not altered.
16. At t=16, Alice sends the ACK

17. At t=17, Alice compares Result2 with Result1. If they match, Alice concludes that Bob knows the password. So Bob is authenticated by Alice.
18. At t=18, Alice send Bob an "Authentication Success" message indicating a successful authentication. Bob is now associated to Alice.
19. At t=19, Bob replies with an Acknowledge.

Figure 5 shows the Authentication between a station S and an Access Point AP. Note that messages in brackets are encrypted

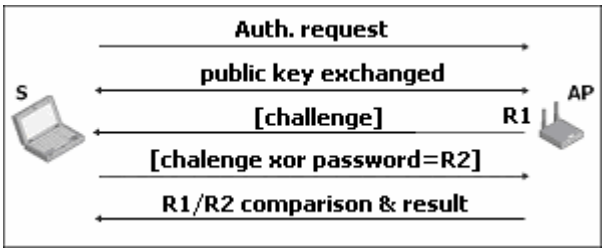

Figure 5 – Authentication process

6. CONCLUSION

Our Hybrid Cryptosystem security specification is a major enhancement over the original WEP. The major component of WEP, that is, the use of the symmetric cipher RC4 to encrypt every frame has not been changed; however, our system adds a series of corrective tools around it to make it more secure.

First, the use of 128-bit WEP key is now mandatory. Moreover, an MD5 hash function was added before the use of any key in RC4, this hash makes sure that no two frames are encrypted using the same Shared Key part as it is used to be in WEP. Every frame has a totally different symmetric key to be used for encryption and decryption.

Second, a Key Management scheme was added to solve the problem of static Shared Keys: even though we are hashing before using the key in RC4, but changing the Shared Key is still needed.

Finally, an authentication mechanism was included: it is based on an encrypted challenge with an encrypted challenge reply. The challenge and its corresponding challenge response are also digitally signed.

REFERENCES

[1] F. Bulk. "The ABCs of 802.11i Wireless LAN Security." Courtesy of Network Computing. 2006.

[2] W. Dai. "Crypto++ 5.2.1 Library." Retrieved January 12, 2005 from W. Dai's personal web site: http://www.eskimo.com/~weidai/cryptlib.html. 2005.

[3] J. A. Davies and W. L. Price. The Application of Digital Signatures Based on Public-Key Cryptosystems. NPL Report DNACS 39/80. National Physics Laboratory. Teddington, England. 1980

[4] T. ElGamal. "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms." IEEE Transactions on Information Theory IT-31, pp. 469-472, 1985.

[5] S. Fluhrer, I. Mantin, and A. Shamir. "Weaknesses in the Key Scheduling Algorithm of RC4." In Eighth Annual Workshop on Selected Areas in Cryptography. Toronto, Canada, 2001.

[6] M. Gast. 802.11 Wireless Networks: The Definitive Guide. O'reilly, 2005.

[7] O. Masafumi. On Operation of 802.11 Wireless Network Services, 2003.

[8] NESSIE Consortium. Portfolio of Recommended Cryptographic Primitives. Retrieved February 6, 2006 from Cryptnessie Web site: http://www.cryptnessie.org, 2003.

[9] G. Ou. Real Wireless LAN security, Introducing 802.1x and EAP. TechRepublic online magazine, CNET network. Retrieved February 2, 2006 from TechRepublic Web site: http://www.techrepublic.com, 2003.

[10] T. Rappaport. Wireless Communications Principles and Practices, 2nd edition. Prentice Hall, 2002.

[11] R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, MIT and RSA Data Security. Web site: http://www.rfc-editor.org/rfc/rfc1321.txt, 1992.

[12] R. Rivest, A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public Key Cryptosystems." Communications of the ACM 21, pp. 120-126, 1978.

[13] B. Schneier. *Applied Cryptography 2nd edition*. John Wiley and Sons, Inc., 1996.

[14] A. Stubblefield, J. Ioannidis, and A. Rubin. Using the Fluhrer, Mantin and Shamir Attack to Break WEP. AT&T Labs – Research. Florham Park, NJ, 2001.

Acknowledgement: This work was funded by the Lebanese American University.