Enhanced Wired Equivalent Privacy for IEEE 802.11 Wireless LANs*

Taejoon Park, Haining Wang, Min-gyu Cho, and Kang G. Shin

Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, MI 48109-2122
{taejoonp,hxw,mgcho,kgshin}@eecs.umich.edu

Abstract

The Wired Equivalent Privacy (WEP) is defined as part of the IEEE 802.11 standard to provide secure communication over a wireless channel. However, it suffers serious security flaws, such as the vulnerability of RC4 to keystream reuse and the misuse of CRC checksum in ensuring data integrity. In this paper, we design, implement, and evaluate a software (middleware) approach, which runs on top of WEP, to fill the security holes of WEP. The core of this middleware is a novel key-management protocol in which (1) to minimize the possibility of keystream reuse, the message-keys for data encryption are frequently refreshed; (2) to achieve secure exchange of message-keys, we append the Hash Message Authentication Code (HMAC) to each message-key, and then encrypt it with a base-key[1]; (3) to provide reliable key-management service, we set up a hidden TCP connection and develop the corresponding daemons at the access point (AP) and a mobile node; and (4) to provide a mobile node with a new base-key and a message-key when the node moves from one *microcell*[2] to another, we realize "secure roaming" based on Inter-Access Point Protocol (IAPP) [2]. Furthermore, to ensure data integrity at the data-link layer, each data frame is augmented with HMAC.

By setting the key-refreshment interval judiciously, we can reduce the rate of keystream reuse to an acceptably low level. More importantly, any compromised message-key becomes unusable after a single refreshment interval, and it does not reveal any information about the future message-keys. Our performance evaluation shows that the computation overhead of the proposed scheme is insignificant even when data is transferred at the maximum rate, and it is feasible for an AP to maintain hidden TCP connections for many mobile nodes.

Keywords— Wired Equivalent Privacy (WEP), middleware, key management, performance evaluation.

1 Introduction

Wireless networks have the inherent weaknesses of security. Unlike the wired counterpart such as Ethernet, wireless networks do not require any physical contact for communications. Within the transmission range, anyone with a simple radio receiver can hear all the conversations over a wireless channel. It is, therefore, much easier to eavesdrop conversations in a wireless network than its wired counterpart.

The IEEE 802.11 [1] specifies the Wired Equivalent Privacy (WEP) encapsulation of 802.11 data frames to provide confidentiality, access control, and data integrity. WEP appends a 32-bit CRC checksum to each outgoing frame, and then encrypts it with RC4 using 40- or 104-bit message-key plus

*The work reported in this paper is supported in part by the DARPA Networked Embedded Systems Technology Program.

[1]The base-key is established through Authentication and Key Exchange (AKE) when a mobile node joins the network.

[2]The microcells are similar to the cells in the cellular telephone system. Each AP controls the microcell.

a 24-bit random *initialization vector* (IV). WEP also decrypts each incoming frame with the message-key, and validates the CRC checksum. In this protocol, RC4 and the shared message-key are used for confidentiality and access control, and CRC checksum for data integrity.

However, WEP has been shown in [7, 24] to fail to meet its design goals. Use of RC4 for data privacy and CRC-32 for data integrity is due mainly to their speed and ease of implementation, but they do not provide cryptographically-secure encryption and authentication. Many serious security flaws have been exposed in these two design choices. These security flaws can invite a number of practical attacks, ranging from eavesdropping to tampering with the transferred data frames.

The weakness of RC4 in WEP is not due to its short message-key length, but its unsound usage of cryptography. It is infeasible to achieve privacy with the WEP encapsulation by simply increasing message-key size from 40 to 104 bits. Attacks against WEP that succeed regardless of the message-key size were reported in [7, 17, 24]. It is well-known that the stream cipher like RC4 is vulnerable to keystream reuse, *i.e.*, it is unsafe to use the same key more than once. Unfortunately, WEP does not provide any automated mechanism to frequently change the message-key, and the message-key is updated manually in the order of days, weeks or even months. To overcome this keystream reuse problem, WEP selects a new 24-bit IV for every frame, to generate a keystream[3] for encryption. However, when WEP randomly selects the IV from the space of $n = 2^{24}$, the probability of collision among IVs exceeds 50 % after only $4823 \approx 2^{12}$ data frames. With the collision probability of 0.00000001, WEP may begin to reuse IVs only after transmission of 6 data frames, implying that an attack can succeed, with a non-negligible probability, in less than a second of transmission. The globally-shared message-key makes IV collisions even more likely, since the chance of IV collisions increases proportionally to the number of users. Hence, an attacker can easily build up a dictionary of keystreams for every IV. Note that an IV is transmitted with each data frame in plaintext.

Another source of the insecurity of WEP is using CRC checksum for data integrity. The CRC checksum was originally designed to detect random errors in messages, not to provide cryptographically-secure authentication code. Moreover, because WEP checksum is a linear[4] and unkeyed operation, an attacker can arbitrarily modify the message without full knowledge of the original message. Thus, the CRC checksum is insufficient to ensure data integrity under malicious attacks. Secure access control is not guaranteed either, thus failing the shared-key authentication mechanism of WEP; attackers could trick the access point to decrypt ciphertexts for them.

In this paper, we propose an enhanced WEP protocol that achieves high-level security, while incurring as small an overhead (in both delay and computation) as possible. The heart of the proposed protocol is to provide an automated mechanism that refreshes the message-key seamlessly. In particular, the proposed protocol is comprised of the following key operations. First, to address the keystream reuse problem, an Access Point (AP) establishes and maintains a *hidden* TCP connection for a legitimate mobile node, through which it periodically provides a new message-key. Since a node cannot communicate with others without the message-key, it must register itself at the AP, by performing the *authentication and key establishment*[5] (AKE) protocol and then setting up a hidden connection. Second, both the AP and mobile nodes use dual key slots in order to renew the message-key, thus unaffecting the ongoing data-frame transmissions. That is, while the key of one slot is being used for data encryption/decryption, the key of the other slot will be updated. It is the AP's responsibility to control when to update or activate a slot. Finally, to address the scalability issue, we present a secure roaming protocol for the proposed key-management, by which each mobile node can securely and seamlessly move from one cell to another.

We realize this protocol with middleware on top of the currently-available WEP implementations.

[3]The generated keystream $RC(v, k)$ is a function of IV, v, and the message-key, k.

[4]$c(x \oplus y) = c(x) \oplus c(y)$ holds for all x and y.

[5]The (entity) authentication is the service that enables communication partners to verify the identity of their peer entities. A successful authentication will lead to the establishment of the shared secret-key.

While minimizing the modification to the existing WEP implementations, we have removed most of the known security flaws in WEP. The middleware assumes a certification authority that issues node's own master secret, and has an external module for the AKE protocol. The core of the middleware is a user-level key-management mechanism in which: (1) to minimize the possibility of keystream reuse, we frequently refresh the message-keys that encrypt the data frames; (2) to securely exchange the message-keys, we append HMAC to each message-key, and then, encrypt it with a base-key; (3) to provide reliable key-management service, we create a hidden TCP connection and develop the corresponding daemons at the AP and a mobile node; and (4) to provide a mobile node with a new base-key and a message-key when the node moves from one *microcell* to another, we realize "secure roaming" based on IAPP. Furthermore, to ensure data integrity at the data-link layer, each data frame is augmented with HMAC.

The remainder of the paper is organized as follows. Section 2 gives a brief review of WEP and known attacks. Section 3 presents the related work. Section 4 describes our proposed protocol. Section 5 describes the software architecture for the proposed middleware. Section 6 presents the results of our performance evaluation for the proposed protocol. Section 7 compares the enhanced WEP with the Extensible Authentication Protocol (EAP), Lightweight Extensible Authentication Protocol (LEAP) and Temporal Key Integrity Protocol (TKIP). Finally, the paper concludes with Section 8.

2 Overview of WEP and Known Attacks

2.1 Overview of WEP

The WEP protocol is intended for secure layer-2 data transmission in an 802.11 wireless LAN environment. It has been widely-implemented by manufacturers into their wireless hardware. For completeness, we give a brief description of the WEP protocol. Its key features include: (1) message-keys are shared between all members of the wireless network; (2) the well-known stream cipher RC4 is used to encrypt the body of the transmitted data frame; and (3) the CRC-32 checksum is used for ensuring data integrity.

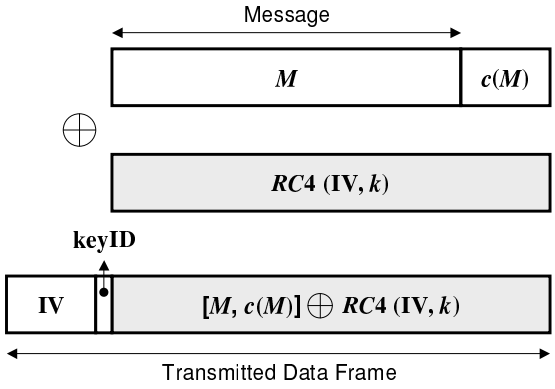

Figure 1: The structure of WEP frame

The encryption and decryption of a data frame are detailed as follows. The data encryption at the sender side proceeds as shown in Figure 1. First, based on the data frame M, we compute its CRC $c(M)$ and attach it at the end of the data frame, *i.e.*, the plaintext is $P = [M, c(M)]$. The plaintext P is then fed as input to the second phase of data encryption. In the second phase, the plaintext P is encrypted by using the stream cipher RC4: (1) a 24-bit IV, v, is randomly selected; (2) a secret key, k, for the encryption is chosen; (3) RC4 generates a keystream, denoted by $RC4(v, k)$, as a function of

v and k; and (4) the plaintext P is XORed with $RC4(v,k)$ to produce the ciphertext. Finally, the IV, the ID of the key slot that stores k (keyID), and the ciphertext are transmitted over the wireless link.

The data decryption at the receiver side simply reverses the encryption process. First, the keystream $RC4(v,k)$ is re-generated using the secret key k referenced by keyID, and then XORed with the ciphertext to recover the plaintext P'. The receiver then checks the data integrity of the recovered plaintext P'. By splitting P' into $[M', c']$, the receiver computes the CRC checksum $c(M')$ of M', and then compares $c(M')$ with the received CRC checksum c'. Only a match will lead to the acceptance of the received data frame.

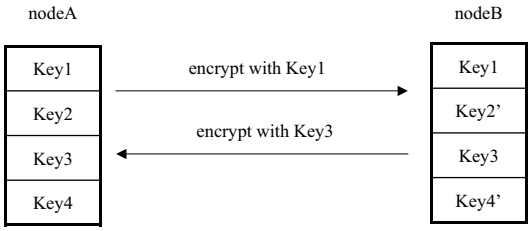

Figure 2: WEP key

At wireless LAN cards, four key slots are available for storing the message-keys, and the one currently being used is referenced by the keyID. For successful communication between two nodes, they should have at least one common key stored in the same key slot. However, if they have two or more common keys, the keyID for encryption may be different in these two nodes, *i.e.*, each node may use the different key stored at the different key slot for data encryption. The receiving node can correctly decrypt the ciphertext by referring to the keyID in the received message, although this keyID may differ from the one for its own data encryption. Figure 2 illustrates this. Nodes A and B have two common keys, say *Key1* and *Key3*. The two nodes can communicate with either of these two keys, and they do not have to use the same key for data encryption. As shown in the figure, node A may encrypt data with *Key1* while node B encrypts data with *Key3*. Node B can correctly decrypt the message from A by accessing key slot 1, while node A can read the encrypted message from B by referring to key slot 3. We utilize this flexibility to achieve the seamless message-key updates, which will be shown in 4.3.

2.2 Known Attacks

As a result of its mis-application of cryptography, WEP is vulnerable to a number of passive and active attacks which are summarized as follows. First, *eavesdropping attack*: by exploiting the RC4's weakness of keystream reuse, the attacker can overhear the conversation between two mobile hosts. Once two encrypted frames with the same IV are discovered, there are many ways for the attacker to figure out the plaintext in a conversation. Second, *decryption dictionary attack*: after obtaining keystreams that are used in encryption, the attacker can build a table of keystreams that covers the entire IV space. Using the table, the attacker can easily decrypt the intercepted ciphertext. Third, since WEP uses the CRC checksum for data integrity and message authentication, several serious attacks can be mounted: (a) message modification: the transmitted message can be modified without being caught; (b) message injection: no secure access control is provided; (c) authentication spoofing: after intercepting an authentication sequence, the attacker can "authenticate" itself without the message-key; and (d) message decryption: the attacker can deceive the AP to decrypt the ciphertext for him. The authors of [7] detailed these possible attacks along with the feasibility of mounting the attacks in practice.

In this paper, we assume that an attacker can forge, modify, and delete any information. The attacker can also mount off-line dictionary attacks for future break-ins, man-in-the-middle attacks, or

session-hijacking. We further assume that a given level of security can be preserved from the attacker only for a certain period of time. These assumptions on the power of an attacker are reasonable in view of today's hardware and firmware technology. The necessary hardware to monitor and inject arbitrary traffic is available to attackers in the form of wireless LAN cards. Active attacks can also be mounted by modifying the firmware of the card to allow injection of arbitrary data frames. Attackers can even read the secret keys stored in the card by using existing tools (*i.e.*, `iwconfig` tools), or by writing and executing a simple program. It is prudent to assume that a "determined" attacker has full access to the link layer for passive and active attacks.

3 Related Work

To enhance the security of WEP, one may replace RC4 with different cryptographic systems. For instance, the author of [24] suggested the use of Advanced Encryption Standard (AES) [10] in Offset Codebook Mode (OCB) [21], a session key derived from the message-key, and a 128-bit IV. However, this is not preferable because (i) it will not work until AES becomes available; and (ii) it is not interoperable with legacy devices. A better approach is to improve the security of WEP without replacing the RC4 encryption. The authors of [8] suggested frequent refreshment of the message-key through mutual authentication. This will make it harder for attackers to compromise the security system. However, it has its own drawbacks, such as the significant delay, high computation and bandwidth requirements that result from frequent 3-way authentications. A better alternative is to refresh the key without frequent authentications.

Besides WEP, another link-layer mechanism for providing authenticated access control is the MAC-level filtering [1]: when a mobile host registers its wireless LAN card, its MAC address is recorded at the AP, thus allowing data frames to and from that MAC to be filtered into the wireless LAN. However, the MAC-level filtering does not provide any confidentiality in data communication, so it is easy to be broken in and difficult to manage.

The most recent layer-2 authentication mechanism is the Robust Security Network (RSN) for the IEEE 802.1x standard's port-based network access control [3], which specifies how EAP [6] can be encapsulated in the Ethernet frames. The EAP is a general protocol for authentication which supports multiple authentication mechanisms. However, as pointed out in [18], the RSN and its EAP authentication suffer from man-in-the-middle attacks and session hijacking, due to their lack of per-packet authenticity and integrity.

LEAP [4] has been proposed by Cisco to support mutual authentication between a mobile host and the AP, and per-session WEP keys. LEAP also defeats man-in-the-middle attacks, which exist in the conventional EAP, through mutual authentication. In LEAP, the mobile host and the AP mutually authenticate each other during the start phase. Then, the AP relays EAP messages to the authentication server using the Remote Authentication Dial-in User Service (RADIUS) protocol [20] during the authentication phase. During the finish phase, the per-session key is transmitted to the mobile host when it is properly authenticated. This authentication can be performed once every 30 minutes or so to prevent the reuse of the same IV. However, this scheme requires the dedicated authentication server and APs capable of speaking the LEAP protocol.

TKIP [12, 13], proposed by RSA Laboratory and Cisco, changes the message-key on a per-frame basis. Under TKIP, a temporal key (TK) is shared among mobile nodes, and the message-key is generated from a hashing function whose input is the tuple <IV, node ID, TK>. It is required that each IV not be used more than once with each TK. Therefore, there will be no keystream reuse problem if such a requirement can be strictly met. Moreover, TKIP adds the message integrity check (MIC) to each WEP frame to augment ineffective data integrity. The disadvantage of TKIP is that it needs firmware upgrades at each mobile client's wireless LAN card.

The message-key management in our enhanced WEP is similar to the group-key management in secure group communications, where a symmetric group-key is known only to group members. Like the AP, the key server controls the access to the group-key: it sends the group-key to new authorized users as well as performs group re-keying. The simplest solution to group re-keying is to use the Group-Key Management Protocol (GKMP) [11], which distributes group-keys through unicast — the re-keying involves a pair-wise secure exchange of the group-key between a central key server and each group member. To reduce the re-keying overhead for single membership changes, efficient re-keying algorithms, such as a logical key tree hierarchy [25] and a boolean minimization techniques [9], have been proposed. Moreover, to improve efficiency further and alleviate the out-of-sync problem, periodic batch re-keying [9, 16, 22] is proposed, which is similar to our proposed periodic key-refreshment.

However, there are several differences between these two key-management problems, which simplify the design of our message-key management. First, mobile hosts within the same microcell share the same medium, and have almost same RTTs between mobile hosts and the AP. So, their link bandwidths and RTTs are homogeneous, instead of being heterogeneous in secure group communications. Second, the number of mobile hosts that an AP can support is limited, but the members in a secure communication group may range from hundreds to millions, so the scalability within a microcell is less crucial to the success of message-key management than that of the batch re-keying in secure group communications. Third, the message-key update requires higher reliability, but looser timeliness than the group re-keying, due to the high loss rate in the wireless environment and the flexible key-slot usage in wireless LAN cards. Therefore, we simply employ TCP unicast for carrying message-key updates, instead of reliable IP multicast used in secure group communications.

4 Enhanced WEP Protocol

The enhanced WEP protocol aims to remove most of the known security flaws of WEP, while incurring as small an overhead (both delay and computation) as possible. Thus, the enhanced WEP meets the following security goals, none of which the original WEP was able to achieve: (1) *confidentiality*: prevents a session from eavesdropping; (2) *data integrity*: prevents tampering with the transmitted messages; (3) *access control*: protects and controls access to a wireless network.

The intrusion model mentioned in Section 2.2 calls for a *renewable* security system. The enhanced WEP protocol realizes this renewability through a key distribution infrastructure comprised of multiple layers, and uses multiple keys as follows:

- *Message-key* (MK): is used to encrypt data frames being transferred;
- *Base-key* (BK): is used to encrypt message-keys and to compute the message authentication codes;
- *Authentication-key* (AK): is used to establish the base-key through authentication.

The IEEE 802.11 specifies two modes of operation: *infrastructure* and *ad hoc* modes. In the infrastructure mode, each mobile node sends data frames to the AP, which then relays them to the destination node. In the ad hoc mode, each node communicates directly with other nodes. The enhanced WEP protocol uses the AP as the key server for MK, and hence, it works best for the infrastructure mode. However, we do not preclude the need for the ad hoc mode, and the proposed protocol supports direct communications between two nodes without the AP's intervention.

The failure of WEP in providing data confidentiality stems from its high probability of keystream reuse. To minimize the probability of keystream reuse while retaining the RC4-based encryption/decryption, we propose a key-management layer that consists of two main mechanisms:

- *Automated key-refreshment*: the AP periodically updates MK and transmits its encrypted version to all legitimate nodes.

- *Seamless data transmission*: the MK refreshment is transparent to the actual data transmission.

The key-management layer creates a "hidden" TCP connection solely for MK refreshment between the AP and a mobile node. This TCP connection is responsible for the reliable MK refreshments. The key-management layer ensures that data transmission will not be disrupted by frequent MK refreshments.

The number of TCP connections that an AP has to manage is small because of the relatively small area coverage of a microcell. As the network size grows or the total number of nodes in the network increases, more microcells will be added, thus keeping the workload of each AP at a reasonable level. In a large wireless LAN (WLAN) consisting of multiple microcells, clients are likely to "roam" from one microcell to another. We thus develop a secure roaming protocol, by which each mobile node can securely acquire BK and MKs for a new AP when it enters a new microcell in the WLAN.

As mentioned earlier, the CRC checksum of WEP is not cryptographically secure for data integrity. Only the keyed (hashed) message authentication code (HMAC) can provide sufficient security to assure the integrity of the received messages. Thus, we append the HMAC to data frames in addition to the CRC checksum.

In this section, we first describe the key-distribution infrastructure, then present the key management, and finally detail the secure roaming and the keyed-message authentication.

4.1 Key-Distribution Infrastructure

The shared-key authentication of WEP[1] is vulnerable to attacks: it is possible for an attacker to inject properly encrypted authentication messages without MK. Thus, we introduce the key-distribution infrastructure based on certificates, while disfavoring the conventional WEP authentication. It is well-known that security goals can be met via valid certificates, encapsulating the AK, issued by a globally-trusted certificate authority (CA).

Since keys can be kept secret only for a limited period of time, BK itself must be negotiable and renewable. The authentication and key exchange protocol is used for this purpose. One can use any authentication protocol as long as it supports mutual authentication. As mentioned in [18], the absence of mutual authentication can be exploited to mount man-in-the-middle attacks.

The key-distribution infrastructure is comprised of 4 sub-layers: certification management, authentication and key exchange, message-key refreshment, and the message encryption/authentication. Each of these sub-layers is elaborated on as follows.

- *Certification management*: provides services such as certificate issuing, renewal and revocation. The certificate provides an AK to each mobile node. A compromised AK is valid only until the certificate is renewed. In infrastructure-based WLANs, we prefer a centralized CA system. However, note that in ad hoc WLANs, if we still rely on a centralized CA to provide certification services, the maintenance of such a server becomes very complicated, due mainly to node mobility and channel unreliability. In such a case, *distributed* certification authorities [14, 23] is a better choice.

- *Authentication and Key Establishment* (AKE): is performed, when a mobile node joins the network, to (1) agree on mutual authenticity between the AP and itself, and then (2) establish the BK shared between the AP and the mobile. AKE can limit the scope of attacks, because the compromised BK can be re-negotiated or replaced, whenever necessary, by re-running the AKE. The AP uses different BKs for different nodes under its control, and hence, attacks to the BK can victimize only the node associated with the BK, without affecting the other nodes.

- *Message-Key Refreshment* (MKR): is initiated by the AP to periodically transmit a new MK encrypted with BK to each legitimate node through a hidden TCP connection. Creation of this connection is preceded by the AKE procedure between the node and the AP. By MKR,

any successful attack will become futile upon refreshing the key, hence achieving the goal of both confidentiality and access control. Furthermore, the number of IV collisions can be made reasonably small by shortening the key-refreshment interval (at the expense of increased overhead).

- *Message Encryption and Authentication* (MEA): uses the currently valid MK to encrypt data frames and compute the corresponding message authentication codes. The encryption engine employs the conventional RC4 cipher. For message authentication, we append HMAC to the plaintext.

4.2 Concept of Key Management

The key-management layer: (1) performs AKE and establishes a hidden TCP connection for each mobile node and (2) time-synchronizes MKR and MEA during the MK refreshment. Figure 3 illustrates how the key-management layer works. In terms of the control (or hidden TCP) connection, node A acts as the key server providing a new MK, and node B functions as the client under the control of A. In case of AP-based communication, the AP plays node A's role and mobiles in the corresponding microcell become node B. While, in the ad hoc mode, a subset of nodes, called *coordinators*, should be elected as key servers (playing the node A's role). Here we only show uni-directional data transmission for the sake of clarity, but our protocol can be easily adapted to the bi-directional case by full-duplexing the MEA operation.

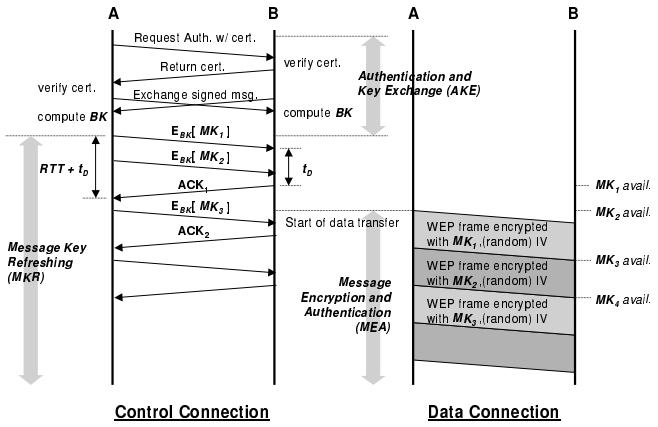

Figure 3: The Concept of key management

Nodes A and B perform AKE in order to prove authenticity and to establish a BK, either when node B joins the network or when they want to renew BK. Figure 3 is based on the assumption of the Diffie-Hellman type of mutual authentication, which is known to protect against man-in-the-middle attacks. A successful AKE leads to the creation of the control connection, through which MK updates are transported.

MKR relies on the MK buffering to relax the time constraint of MK refreshment: node A transmits MK to node B in advance, so that node B has enough time to decrypt and store the MK in the corresponding key-slot. Periodically, node A transmits N control packets, encrypted with BK, each of which contains MK_i, $i = 1, \cdots, N$. After receiving the acknowledgment (ACK) for MK_i from B, A and B are ready to use MK_i for data encryption/decryption. To handle packet losses in the control connection, it must ensure transmission reliability. So, we use TCP for carrying control packets.

The key management only requires very loose timeliness thanks to the MK buffering. So, the system can tolerate spontaneous large round-trip delays due to packet losses. The number, N, of MKs

transmitted beforehand dictates its robustness to transmission failures. The larger N, the looser the timeliness requirement. The choice of $N = 1$ works well if the interval of MK refreshment is in the order of a few seconds.

The key management incurs an initial latency of $RTT + t_D$, where t_D is the decryption time at node B. During that period, the data connection is temporarily disabled, but this happens only when the BK is renewed.

Our proposed scheme does not require any modification to the IV selection procedure. Hence, the wireless LAN cards may randomly choose a per-frame IV, resulting in a non-negligible probability of IV collision. However, by setting the key-refreshment interval properly, the probability of IV collision can be lowered to the desired level. Most importantly, dictionary attacks, either on-line or off-line, on MKs are no longer possible, because a compromised MK becomes invalid upon renewal of MK.

4.3 Realization of Key-Management Layer

We now describe how the key-management concept can be realized within the framework of WEP specification. To make the key-management protocol compatible with the existing widely-deployed WEP-enabled mobile devices, it should not require any modification to the wireless LAN cards, their firmware, and device drivers. This requirement is difficult to meet, since the WEP protocol requires the same key present in the matching key slots of different wireless LAN cards for proper encryption/decryption. The design challenge can be re-stated as: "how can all nodes (1) securely receive a new MK and (2) switch to the new MK without disrupting the ongoing data transmission?"

The problem of distributing a new MK is resolved by introducing "hidden" TCP connections. The AP manages these TCP connections with a dedicated port number. It periodically encrypts with BK and transmits to all mobile nodes the following information: a new MK, a keyID that the MK should be written to, a timestamp, t_w, indicating when the MK should be written to the key-slot keyID, and another timestamp, t_a, indicating when the keyID should be activated. Figure 4 depicts the format of the control packet. Note that control packets include the HMAC computed using a key derived from BK. This ensures the integrity of packets and protects them against active attacks.

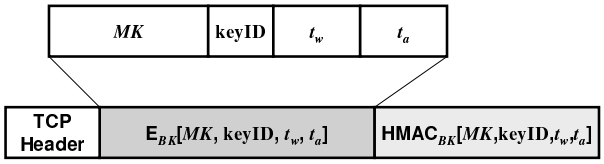

Figure 4: A control packet with HMAC in the enhanced WEP

The key idea to seamlessly refresh the MK is to use two key slots at each node in parallel. While one key slot is active and its key is being used for data encryption/decryption, the new MK is written to the other slot at time t_w. Then, at a later time t_a, each node switches the active key slot to the one with the new MK. The salient feature of this update scheme is that both current and future keys are present in the matching key slots of all nodes, thereby ensuring proper decryption of data traffic.

Let $T_{refresh}$ denote the key-refreshment interval and t_{start} be the time when the AP starts to refresh MK. The setting of t_w is fairly flexible, as long as it allows each node to have enough time to receive and decrypt the MK.[6] Then, we set t_a to $t_w + \frac{T_{refresh}}{2}$ so that each node has enough time to write the MK into the key-slot.

[6]The range of t_w is given as: $t_{start} + \frac{T_{refresh}}{2} \leq t_w \leq t_{start} + \left(N + \frac{1}{2}\right) \cdot T_{refresh}$.

Each control packet creates two associations, $[t_w, MK, keyID]$ and $[t_a, keyID]$. The former controls when to write MK to the key-slot keyID, and the latter indicates when to activate the MK in the slot keyID. Figure 5 illustrates, in the time domain, how key slots are updated. The $(2i+1)^{\text{th}}$ control packet creates associations, $[t_{w,2i+1}, MK_{2i+1}, odd\_key]$ and $[t_{a,2i+1}, odd\_key]$, and causes the following two events to occur. At time $t_{w,2i+1}$, the node writes MK_{2i+1} to the key-slot, odd_key. At a later time $t_{a,2i+1}$, it switches its active key slot to odd_key, which contains MK_{2i+1}. Note that $t_{w,2i+1} + T_{refresh}$ is the hard deadline for activating MK_{2i+1}. Likewise, subsequent arrivals of control packets, $(2i+2)^{\text{th}}, (2i+3)^{\text{th}}, \cdots$, will be processed. Hence, events for writing the MK and those for switching active key slots are processed concurrently under the AP's control.

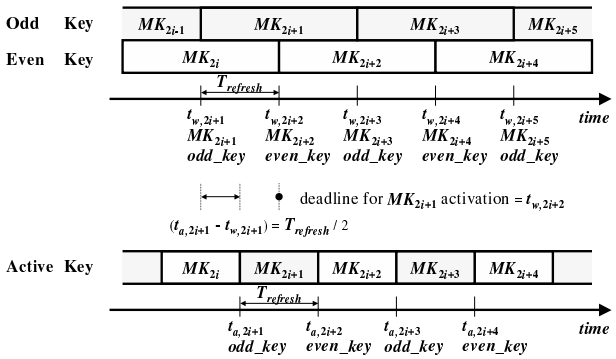

Figure 5: Timing diagram

The proposed key-management is robust against clock skews among nodes. As an example, Figure 6 illustrates, in the time domain, key slots of two nodes, K and L, when there exists a clock skew. Let c_j be the mapping from clock time to real time at node j, where $c_j(T) = t$ means that at clock time T the real time is t. Then, the clock skew between K and L is given by $\delta = |c_K(T) - c_L(T)|$. The figure shows that seamless MK refreshment is preserved, if $\delta < \frac{T_{refresh}}{2}$. When $c_K(t_{a,2i+1}) < t < c_L(t_{a,2i+1})$, node K uses MK_{2i+1} for encryption, while node L still uses MK_{2i} due to the clock skew. But, since both K and L have the same decryption key pair, $\{MK_{2i}, MK_{2i+1}\}$, during this period, they can communicate with each other. In general, the worst-case clock skew must be less than a half of the MK refreshment interval, *i.e.*, $\max\{|c_K(t_a) - c_L(t_a)| : \forall K, L\} < \frac{T_{refresh}}{2}$. As shown in Section 6.1, $T_{refresh}$ is in the order of seconds. Therefore, this timing constraint is very loose and easy to meet.

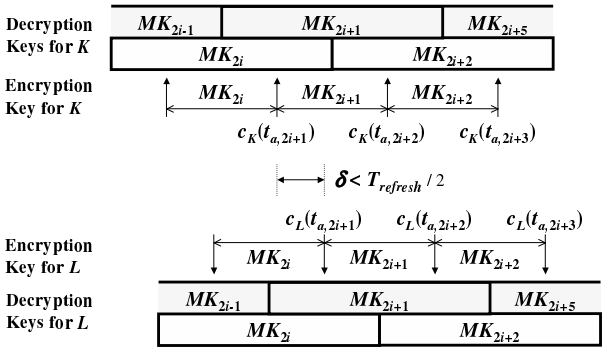

Figure 6: The clock skew

Figure 7 shows the proposed key-management layer that realizes the above-mentioned schemes. The WEP protocol specifies wireless LAN cards to be able to store up to 4 MKs, one of which can be chosen

for encryption/decryption. As mentioned earlier, both the AP and a mobile node cooperatively perform an AKE protocol to verify mutual authenticity. As part of the AKE protocol, they establish BK.

After AKE, the AP periodically distributes a new MK to all legitimate nodes. In Figure 7, N is set to 1. The AP first chooses MK, $keyID$, t_w and t_a, then generates the control packet as shown in Figure 4. The choice of $keyID$, t_w and t_a determines the timing of MK refreshment as shown in Figure 5.

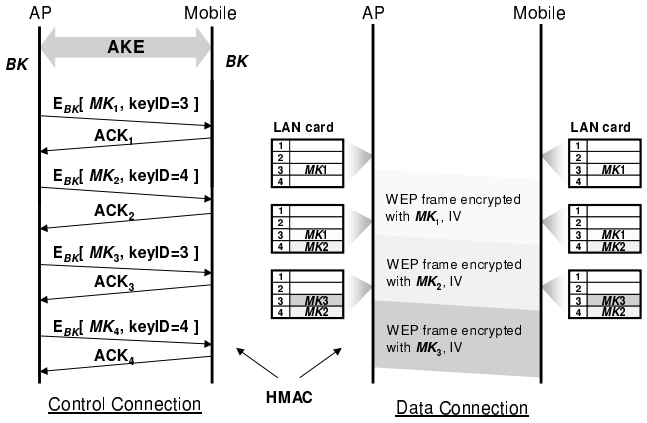

Figure 7: The proposed key-management layer

For each MK refreshment, both the AP and mobile nodes perform the following operations concurrently:

- *Distributing MK*: the AP periodically encrypts and transmits the control packet to all legitimate nodes. Then, the mobile nodes check the integrity of the received control packet through the HMAC test, and, if successful, register two events associated with the received packet. Hence, both the AP and mobile nodes share the same events, [t_w, MK, $keyID$] and [t_a, $keyID$].

- *Writing MK*: the AP updates its $keyID$ slot with the MK at time t_w. Each mobile node, upon receiving the control packet, compares t_w with the current time, t_{curr}, and proceeds as follows: if $t_w + T_{refresh} < t_{curr}$, then it discards [$t_w$,$MK$,keyID]; else if $t_w < t_{curr} < t_w + T_{refresh}$, then it directly writes the MK to the $keyID$ slot; otherwise, it waits until time t_w to write the MK.

- *Activating MK*: the AP switches its active key-slot to the associated $keyID$ at time t_a. Each mobile node, upon receiving the control packet, checks if the current time has not yet passed t_a, and if not, waits until t_a to activate the $keyID$ slot; otherwise, it activates the $keyID$ slot right away.

4.4 Secure Roaming

A large wireless LAN would normally be comprised of multiple microcells, where an AP maintains the security context, BK and MKs, of its microcell. A mobile may roam from one microcell/AP to another. The mobile can roam with software and hardware that preserves a steady network connection by monitoring the signal strength from in-range APs and locking onto the one with best quality. This is called a *hand-off* and is completely transparent to the users.

We use the Inter-Access Point Protocol (IAPP) [2] to achieve secure roaming and to provide the mobile node with new BK and MKs. Figure 8 illustrates how the secure roaming works. We assume that the authentication server (AS) and each AP have verified the identity of each other and established

a shared key. So, AP1 and AP2 have $K_{A,1}$ and $K_{A,2}$, respectively, and the AS stores both $K_{A,1}$ and $K_{A,2}$. In addition, AP1 and AP2 maintain their own enhanced WEP keys, $[BK^{(1)}, MK_{2k}^{(1)}, MK_{2k+1}^{(1)}]$ and $[BK^{(2)}, MK_{2i}^{(2)}, MK_{2i+1}^{(2)}]$, respectively. When AP2 receives a `Reassociate` packet that represents roaming from AP1 to AP2, AP2 sends the AS an `AccessRequest` packet with AP1's ID. Then, the AS responds with an `AccessAccept` packet with: (1) AP1's IP address, (2) the security block[7] for AP1, SB_1, containing a key $K_{1,2}$ encrypted with $K_{A,1}$, and (3) the security block for AP2, SB_2, containing a key $K_{1,2}$ encrypted with $K_{A,2}$. Note that $K_{1,2}$ is the shared key between AP1 and AP2, and will be used for AP–AP communication. Both `AccessRequest` and `AccessAccept` packets are protected by $K_{A,2}$. Next, AP2 sends AP1 SB_1 as a `SendSecurityBlock` packet. AP1 acknowledges it with an `AckSecurityBlock` packet. At this point, both APs have the shared secret $K_{1,2}$ and it will be used to encrypt all future packets. AP2 then sends a `MoveRequest` packet to AP1 and gets back a `MoveResponse` packet containing $BK^{(1)}$ for AP1's microcell. Finally, AP2 encrypts with $BK^{(1)}$ and sends the mobile node a new security context: $BK^{(2)}$, $MK_{2i}^{(2)}$ and $MK_{2i+1}^{(2)}$.

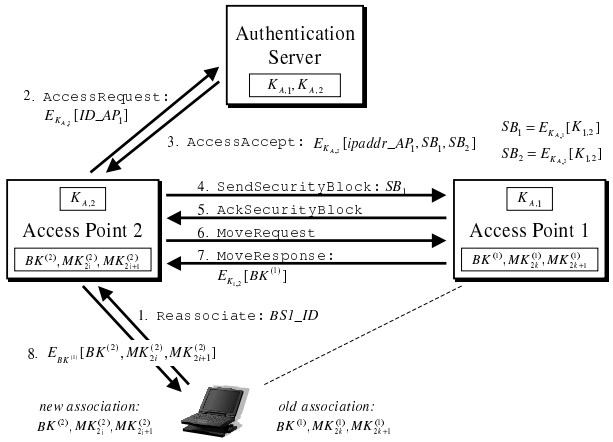

Figure 8: Secure roaming in enhanced WEP

4.5 Keyed Message Authentication

Data traffic is protected from active attacks by the MAC. As shown in Figure 9, we include a HMAC within each data frame, which is computed using a key derived from the MK. We do not require modification to the CRC field, thus preserving compatibility with existing WEP implementations. Note that the use of HMAC degrades the effective throughput, but protects data traffic from active attacks. Message modification/injection attacks that can pass the CRC check, will be rejected by the HMAC test. Note that it is conceptually similar to the Message Integrity Check (MIC) of TKIP [12, 13].

5 Software Architecture

To preserve compatibility with the existing WEP implementations, we implement the enhanced WEP protocol as middleware running atop the current WEP specification.

5.1 Implementation of Key-Management Layer

The key-management layer is comprised of two programs; `ewepd` (*E*nhanced-*WEP D*aemon process) for the AP and `ewep` (*E*nhanced-*WEP* client process) for mobile nodes. Their interactions are based on a

[7] It stores a shared-key encrypted with a secret-key of the recipient.

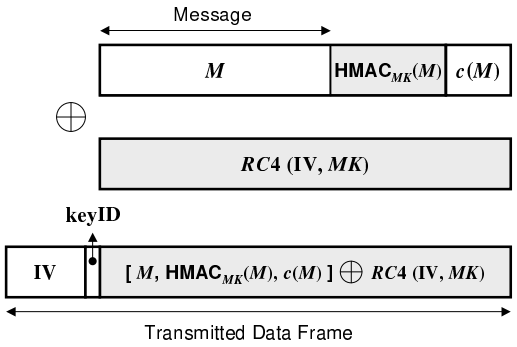

Figure 9: Data frame with HMAC in the enhanced WEP

traditional client-server model. As shown in Figure 10, they cooperatively perform an AKE protocol, exchange MKs, and update their own key slots synchronously. We assume that the AP's IP address and the port number are well-known to `ewep` processes. The `ewep` process opens a TCP connection with the AP when its node joins the network, and closes the TCP connection when the node leaves the network. The `ewepd` process is responsible for bookkeeping hidden TCP connections, and periodically refreshes the MK using these TCP connections. Since the MK is frequently refreshed, a mobile node must register itself to the AP first; otherwise, without the updated MK it cannot securely communicate with other nodes in the wireless LAN. As described earlier, only loose timeliness is required for `ewepd` and `ewep` to update their corresponding key-slot with the same MK.

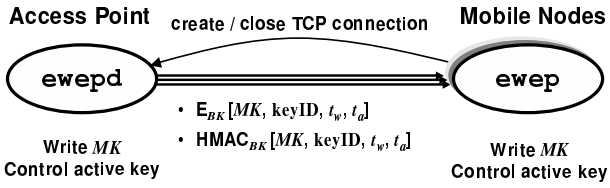

Figure 10: Interaction between swepd and swep

5.2 Implementation of Keyed Message Authentication

For the integrity check on incoming packets, HMAC is appended to every packet. Since HMAC uses a secure hash function such as MD5 or SHA-1 with the shared key, data integrity is ensured. In our implementation, we used MD5 for HMAC, but there is nothing that prevents use of SHA-1 or other secure hash functions.

We modified the device driver of the wireless LAN card to append HMAC to every packet. On the sender's side, HMAC is appended to the IP packets. Whenever an IP packet arrives from the upper layer to the wireless device driver, HMAC with the MK is calculated. This HMAC is appended at the end of the IP packet, and delivered to the device. Thus, the 802.11 payload consists of the IP packet and HMAC as shown in Figure 9.

Only one line of code is modified in the kernel for appending HMAC to IP packets. Different protocol stacks use one buffer for the same packet to reduce extra copies for better performance. When the buffer is allocated for a packet, the buffer size is determined, depending on the packet size plus expected additional headers which will be added during its traversal of protocol stacks. Since we append HMAC at the end of each packet, the buffer is likely to have no extra space for HMAC. Thus, it is necessary to allocate enough space to hold an additional HMAC value. In Linux, the *alloc_skb()*

function is responsible for allocating the buffer, and hence, one line of code in that function is modified to increase the size of buffer to be allocated.

On the receiver's side, only those packets with valid HMAC are delivered to the upper layer. When a packet arrives at the device driver, the HMAC value is calculated with MK, and this calculated value is compared to the received HMAC. When two HMACs match, the packet is considered valid; otherwise, the packet is discarded.

6 Performance Evaluation

The performance evaluation of our proposed approach consists of three parts: (1) how much security improvement it makes; (2) how much overhead one has to pay for the improvement; (3) the feasibility of AP for maintaining a hidden TCP connection per mobile node. Although it is clear that frequent key refreshment significantly reduces the risk of keystream reuse, and HMAC consolidates data integrity and the access control, we need to quantify the improvements and the associated cost. Statistical analyses are conducted to verify the security enhancement. The system overheads incurred by the key-management and the message authentication are evaluated thoroughly. Moreover, we show that it is feasible to support hidden TCP connections in an AP for a large number of mobile nodes. In this section, we first present statistical analyses, then evaluate the corresponding system overhead, and finally, discuss the scalability of an AP when it deals with a large number of mobile nodes in its microcell.

6.1 Statistical Analyses

Due to the key refreshment, the problem of IV collision is limited to a single refreshment interval. The finer-grained key refreshment, the less chance of IV collision but with higher system overhead. To balance security and performance, we need to choose an appropriate key-refreshment interval. We present two criteria to determine an appropriate key-refreshment interval.

The first criterion determines the refreshment interval, given the worst-case probability of IV collision, $p_{collision}$. Based on the birthday paradox, the probability of having at least one IV collision when k IVs are randomly selected among $n = 2^{24}$ IVs is given by

$$P(n,k) = 1 - \frac{n!}{(n-k)! \cdot n^k}.$$

Figure 11 plots the probability of IV collision vs. the number of data frames. Then the criterion is to find the largest k (the number of data frames) such that

$$P(n,k) \leq p_{collision}.$$

We can calculate required key-refreshment intervals for various IV collision rates. Table 1 summarizes the result, where the frame size and the transmission rate are set to 1,500 [Bytes] and 11 [Mbps], respectively.

This criterion guarantees that keystreams be collision-free with confidence probability, $1 - p_{collision}$. However, the resulting refreshment interval gets very short, thus inducing high overhead. For instance, if the key-refreshment interval is 1.5 seconds, the probability of IV collision is reduced to around 5% with 11 [Mpbs] and 6% of CPU time. Moreover, $p_{collision}$ is not directly related to the number of expected IV collisions, and hence, it is difficult to determine the refreshment interval based on the degree of keystream reuse allowed.

We would like to generalize the above criterion so as to quantify the security level and derive the trade-off between security and performance as a function of the security level. A generalized criterion

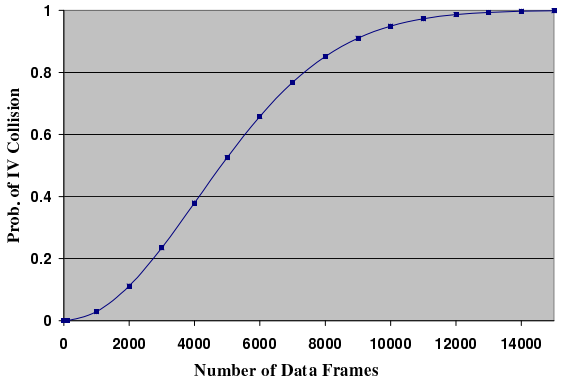

Figure 11: Probability of IV collision

Table 1: Key-refreshment interval

Prob. of IV collision	Key refreshment (s)
1%	0.65
5%	1.47
10%	2.10
15%	2.62
20%	3.07

relaxes the previous requirement of no IV collision, thus allowing a certain number of collisions to occur. The key-refreshment interval is chosen to guarantee that the number of IV collisions be less than $n_{collision}$, where $n_{collision}$ determines the security level.

Kullback [15] provided a formula for predicting the expected number of collisions. For random selection of k IVs, when there are $n = 2^{24}$ different IVs, the average number of IVs occurring exactly r times is given by

$$E_r(n,k,r) = \frac{k(k-1)\cdots(k-r+1)}{r!} \cdot \frac{n(1-1/n)^{k-r}}{n^r}.$$

Since all IVs with $r \geq 2$ should be counted as "one" collision, the expected number of IV collisions is derived as

$$N(n,k) = \sum_{r=2}^{k} E_r(n,k,r).$$

The generalized criterion finds the largest k (the number of data frames) such that

$$N(n,k) \leq n_{collision}.$$

By choosing $n_{collision} \geq 1$ appropriately, one can make a trade-off between security and performance. The larger $n_{collision}$, the lower security level, but with higher performance. The generalized criterion implies that there is a non-negligible probability that an attacker will acquire up to $n_{collision}$ keystreams during each refreshment interval. Since these keystreams become useless after one key-refreshment interval, one is allowed to choose a large $n_{collision}$ value without sacrificing security too much. However, if the highest security level (*i.e.*, 0.000001% collision probability) is required, one must use the former criterion.

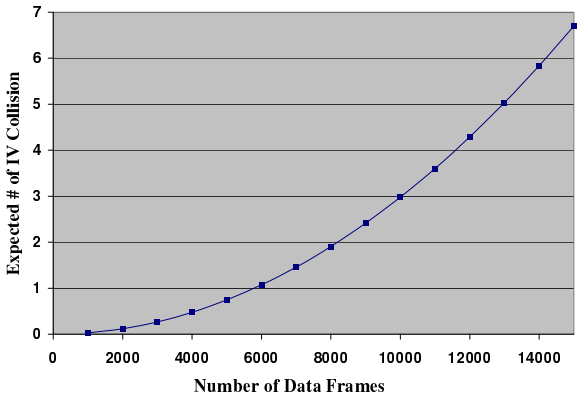

Figure 12: Number of IV collisions

The trade-off between the level of security and the number of frames is shown in Figure 12. The expected number of collisions are shown to be very small even when a large number of frames are transmitted. Thus, one can set the key-refreshment interval to an appropriate value while allowing a small number of collisions. Table 2 shows the required key-refreshment intervals for various values of $n_{collision}$. Clearly, the generalized criterion makes a security trade-off by the choice of $n_{collision}$. For instance, with the key-refreshment interval of 10 seconds, about 3 IV collisions are expected to occur.

Table 2: Key-refreshment interval

Number of IV collision	Key refreshment (s)
2	8.84
3	10.90
4	12.65
5	14.18
6	15.38

Besides the significantly-reduced keystream reuse, the frequent key refreshment also offers the following two important merits: (1) it is robust against dictionary attacks; and (2) the compromised key is valid only within one refreshment interval. Based on the statistical analyses, we set the key-refreshment interval to 10 seconds, which makes a good balance between security and performance. As shown below, the total key-management overhead is less than 100 *msec*, so less than 1% of CPU cycles are consumed by key-management.

6.2 System Overhead

The system overhead is comprised of two parts: (1) the key-management overhead in each key refreshment interval; and (2) the data-link layer authentication overhead for each encrypted data frame. We conducted experiments on the following platform. Two laptops (both the AP and a mobile host) were configured with Intel Pentium III 600 MHz processor, 128 MB RAM and running Redhat Linux 7.1. The WaveLAN cards we used were Orinoco WaveLAN Gold cards with a 128-bit key.

The key-management overhead can be divided further into four parts: encryption/decryption, HMAC, transmission/reception and key-update. The key-management overhead at the AP is listed in Table 3, and that of the mobile host is given in Table 4. They clearly show that the key-update overhead is the dominating factor, which is around 97 *msec* with a high variance at both sides, and more than 99% of the overhead were attributed to key-updates. The overhead of HMAC at both sides

Table 3: Key-management overhead at an Access Point (μ sec)

Encrypt	HMAC	Tranx	KeyUpdate	Total
93	42	230	97,735	99,439
94	42	279	97,843	99,591
94	43	230	97,455	99,182
94	43	228	96,742	98,387
94	43	229	97,026	98,658

are the same — 43 μsec. There is only a 14-μsec difference between encryption and decryption at either side.

Table 4: Key-management overhead at a mobile host (μ sec)

Decrypt	HMAC	Recv	KeyUpdate	Total
108	43	11	97,328	98,357
108	43	10	97,337	98,369
108	43	11	97,146	98,207
107	43	11	96,773	97,802
107	43	11	96,447	98,101

However, the AP's overhead of sending a key is quite different from that of receiving the key at the mobile host. The reason for this lies in the use of the hidden TCP connection: the sending system call does not return until an ACK from the mobile host arrived, whereas the receiving system call is simply copying the content from the socket buffer to the application buffer.

For each data frame, the overhead incurred by HMAC computation is 49 *msec*. Assuming 11 Mbps wireless link capacity and 1,500-byte data frame size, the total overhead introduced at the data-link layer is less than 5% of CPU cycles.

6.3 Feasibility of Maintaining TCP Connections

One concern is the feasibility of maintaining a hidden-TCP connection per mobile node, i.e., the overhead of AP in supporting a large number of mobile nodes. Since all mobile nodes share the same message-key, the only difference between the AP's CPU cycle consumption in supporting single and many nodes lies in the transmission overhead, the rest of overheads are constant. As shown in Table 3, the overwhelming percentage of CPU cycles taken by key-management are used for key-updates, and the total CPU consumption slowly increases with the number of mobile nodes. Figure 13 plots the CPU overhead vs. the number of mobile nodes. Even if there are 1,000 mobile nodes within a microcell, the total CPU overhead of AP is less than 0.4 second for every 10 seconds, i.e., less than 4% CPU cycles are used for key-management. The AP could further reduce its transmission overhead by overlapping the data transmission with the wait for acknowledgment.

Keeping a full TCP connection state requires about 400 bytes of memory in the Linux kernel. To support 1,000 hidden TCP connections, the total memory cost is 400 KB. An AP can meet this requirement very easily without affecting its normal operation. In comparison with TCP snoop [5], the hidden TCP connections consume much less memory space, because only 128 bits are transmitted in 10 seconds among these TCP connections — the buffer requirement at the AP is negligible. By contrast, TCP snoop [5] has to cache all unacknowledged TCP data packets for each TCP connection and keep track of all the acknowledgments.

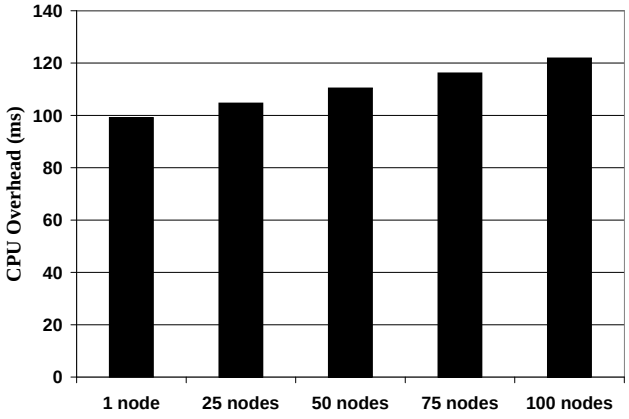

Figure 13: CPU overhead vs. number of mobile nodes

7 Discussion

For the purpose of comparing the enhanced WEP with existing EAP-based protocols (*i.e.*, LEAP protocol), we summarize the strengths of the enhanced WEP over EAP as follows:

- *Efficient re-keying*: in the enhanced WEP, an AP acts as a key server that distributes a new MK, while nodes simply update their own key slots with the MK received from the AP. In contrast, EAP-enabled nodes must perform re-authentication frequently to update the MK. Clearly, the enhanced WEP requires less computation/bandwidth than re-authentication.

- *Seamless re-keying*: thanks to the MK buffering, re-keying in the enhanced WEP does not interfere with ongoing data traffic. To our best knowledge, no previous work supports this feature.

- *Automated re-keying*: other than the initial setup, no human intervention in the re-keying process is needed.

- *Compatibility*: the proposed key-management is compatible with current WEP implementations, kernels, and device drivers. Kernel modification for HMAC support is minimal — only one-line modification is needed.

- *Trade-off between security and performance*: a criterion is derived, determining the re-keying interval based on allowable IV collisions.

- *Robustness against clock skews*: it can sustain clock skews up to a half of the key-refreshment interval.

- *Strong message authentication*: each data frame is augmented with HMAC.

The enhanced WEP scales well with the size of the wireless LAN. First, each AP supports secure and fast roaming for mobile nodes that has already been authenticated with any one of APs in the network. Second, the additional overhead incurred by enhanced WEP will be small, *i.e.*, less than 2% in a 500-node microcell.

Then we compare the performance of the enhanced WEP with that of Cisco's LEAP. The major difference between the two protocols is how to maintain and update MKs. While the enhanced WEP uses the same MK for all nodes, LEAP requires per-session or per-node MKs. Unfortunately, the requirement of multiple MKs causes several problems: (1) it requires a more powerful AP which can

store all MKs and manage a large number of re-authentication requests; (2) with the conventional wireless LAN cards (with 4 key slots), nodes can only communicate with the AP; and (3) inter-node communication will be done via the AP, which has to re-encrypt data frames. On the other hand, the proposed protocol works well with current APs and supports direct node-to-node communication.

Finally, we discuss how the enhanced WEP differs from TKIP. TKIP requires the firmware of all wireless LAN cards to be upgraded in order to implement "fast packet keying," which generates a per-packet RC4 encryption key by hashing MK, IV and node ID of the transmitter. However, this approach has the compatibility and deployment problem with current wireless LANs: for a medium- or large-scale wireless LAN that contains multiple APs, it is costly and cumbersome to upgrade all mobile nodes with TKIP. By contrast, the enhanced WEP does not require the modification on the hardware/firmware of existing wireless LAN cards, and can be easily deployed by upgrading the firmware of APs only.

8 Conclusion

In this paper, we proposed an enhanced WEP protocol that fixes most of the known security flaws of WEP, while preserving the compatibility with existing wireless LAN cards. The main idea is to use the AP as a key server for seamless and frequent message-key updates. The frequent message-key refreshment effectively prevents the occurrence of keystream reuse, and the corresponding authentication secures access control. The core components of the enhanced WEP are: (i) key-distribution infrastructure; (ii) key-management layer; (iii) secure roaming; and (iv) keyed message authentication. Furthermore, HMAC is also employed at the data-link layer, thus ensuring the data integrity of each frame.

We implemented this protocol as middleware atop the current WEP implementation, where the AP uses hidden TCP connections for reliable message-key delivery. Then, we conducted performance evaluation with statistical analysis, system overhead measurements, and the AP scalability analysis. Our analyses and measurements have shown that (i) the enhanced WEP significantly reduces the probability of keystream reuse with small overhead; (ii) it is not difficult for an AP to maintain hidden TCP connections for mobile nodes in its microcell/cluster.

References

[1] IEEE Std 802.11-1997, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*, available from http://www.drizzle.com/~aboba/IEEE/802_11-1997.PDF.

[2] IEEE Std 802.11f/D3, "Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation", *IEEE*, January 2002.

[3] IEEE Draft 802.1x/D1, "Port Based Network Access Control", available from http://www.ieee802.org/1/mirror/8021/docs99/PortNACIEEE.pdf.

[4] "Lightweight Extensible Authentication Protocol (LEAP)", available from http://www.cisco.com/warp/public/102/wlan/nextgen.html.

[5] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP Performance over Wireless Networks", *Proceedings of ACM MOBICOM'95*, Berkeley, CA, November 95.

[6] L. Blunk and J. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)", *RFC 2284*, March 1998.

[7] N. Borisov, I. Goldberg, and D. Wagner, "Intercepting Mobile Communications: The Insecurity of 802.11", *Proceedings of ACM MOBICOM'2001*, Rome, Italy, July 2001.

[8] N. Cam-Winget, J. Walker, B. Aboba, and J. Kubler, "Rapid Re-keying WEP; A Recommended Practice to Improve WLAN Security", IETF, August 2001.

[9] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha, "Key Management for Secure Internet Multicast using Boolean Function Minimization Techniques", *Proceedings of IEEE INFOCOM'99*, New York City, NY, March 1999.

[10] J. Daemon and V. Rijmen, "AES Proposal: Rijndael", available from http://csrc.nist.gov/encryption/aes/.

[11] H. Harney and C. Muchenhirn, "Group Key management Protocol (GKMP) Architecture", *RFC 2094*, July 1997.

[12] R. Housley and D. Whiting, "Temporal Key Hash", *IEEE P802.11 Wireless LANs*, December 2001.

[13] R. Housley and D. Whiting, "Alternate Temporal Key Hash", *IEEE P802.11 Wireless LANs*, April 2002.

[14] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks", *Proceedings of IEEE ICNP'2001*, Riverside, CA, October 2001.

[15] S. Kullback, "Statistical Methods in Cryptanaylsis", Aegean Park Press, 1976.

[16] X. S. Li, Y. R. Yang, M. G. Gouda, and S. S. Lam, "Batch Rekeying for Secure Group Communications" *Proceedings of 10th International World Wide Web Conference*, Hong Kong, China, May 2001.

[17] D. A. McGrew and S. R. Fluhrer, "The Stream Cipher Encapsulating Security Payload", *draft-mcgrew-ipsec-scesp-01.txt*, IETF, July 2000.

[18] A. Mishra and W. A. Arbaugh, "An Initial Security Analysis of the IEEE 802.1X Standard", available from http://www.cs.umd.edu/ waa/1x.pdf, February 2002.

[19] D. Nguyen, L. Zhao, P.-O. Uisawang, and J. Platt, "Security Routing Analysis for Mobile Ad Hoc Networks", available from http://198.11.21.25/capstoneTest/ Students/Papers/docs/Final_revision37165.PDF.

[20] C. Rigney, S. Wilens, A. Rubens, and W. Simpson, "Remote Authentication Dial in User Service (RADIUS)", *RFC 2865*, June 2000.

[21] P. Rogaway, "OCB Mode: Parallelizable Authenticated Encryption", available from http://csrc.nist.gov/encryp-tion/aes/.

[22] S. Setia, S. Koussih, S. Jajodia, and E. Harder, "Kronos: A Scalable Group Re-keying Approach for Secure Multicast", *Proceedings of IEEE Symposium on Security and Privacy'2000*, Oakland, CA, May 2000.

[23] A. Shamir, "How to Share a Secret", *Communications of ACM*, 22(11), 1979.

[24] J. R. Walker, "Unsafe at any Key Size; An Analysis of the WEP Encapsulation", IETF, October 2000.

[25] C. K. Wong, M. G. Gouda, and S. S. Lam, "Secure Group Communications using Key Graphs", *Proceedings of ACM SIGCOMM'98*, Vancouver, Canada, September 1998.