Understanding the difficulties in security protocol design and attempting to relocate the struggle between hacker and defender to a different protocol layer.

SECURITY FLAWS IN 802.11 DATA LINK PROTOCOLS

By Nancy Cam-Winget, Russ Housley, David Wagner, and Jesse Walker

It is evident that anyone with a radio receiver can eavesdrop on a wireless local area network (WLAN), and therefore widely acknowledged that a WLAN needs a mechanism to counter this threat. It is less understood but equally true that anyone with a transmitter can write messages to a WLAN, rendering access controls meaningless. Because forgeries are easy to create, a WLAN needs mechanisms to counter this threat, too.

The IEEE 802.11 standard [7] defines a data confidentiality mechanism known as Wireless Equivalent Privacy (WEP). WEP works using RC4 encryption with a shared key; see the article by Housley and Arbaugh in this issue for more details [6]. The security goal of WEP is data confidentiality equivalent to that of a wired LAN. WEP falls short of this objective,

which is an intuitively appealing but vague security goal. As a consequence, WEP was insufficiently thought through, and numerous flaws quickly appeared. The discovery of shortcomings led to a process to replace WEP by more robust security protocols. Here we review the WEP security flaws and assess the progress made toward replacing WEP.

The study of WEP flaws illustrates the difficulties in security protocol design. An understanding of these flaws helps clarify the choices made by the designers of the new protocols. The IEEE 802.11 Task Group i (TGi) is developing the new WLAN security protocols. The data link security protocols are named TKIP and CCMP. TKIP is a WEP patch, designed for existing hardware. Since TKIP is not ideal, TGi also specifies CCMP to supercede WEP and TKIP. TGi is also defining WLAN authentication and key management enhancements; however, the authentication and key management work is beyond the scope of this article.

Problems With WEP

WEP has several serious inherent problems. It does not meet its fundamental goals of wired-equivalent confidentiality. It also fails to meet the expected goals for integrity and authentication.

WEP has two generic limitations. First, use of WEP is optional, and as a result, many real installations never even turn on encryption. This is unfortunate, as it does not matter how good the cryptography is if it is never used. Second, by default, WEP uses a single shared key common to all users of a WLAN, and this common key is often stored in software-accessible storage on each device. If any device is lost, stolen, or compromised, the only recourse is to change the shared secret in all of the remaining devices. Since WEP does not include a key management protocol, distributing the new secret to all users is an unwieldy process. As a result, key compromises are often ignored. Regardless of the cryptography employed, these shortcomings make it difficult to obtain confidentiality and integrity in WLAN deployments.

In practice, the most serious problem with WEP is its encryption keys can be recovered through cryptanalysis. WEP uses a common stream cipher, RC4, but in a nonstandard way: WEP concatenates a base key with a 24-bit per-packet nonce, called the WEP Initialization Vector (IV), and uses the result as a per-packet RC4 key. In August 2001, Fluhrer, Mantin, and Shamir described a stunning new attack on this construction [5]. They showed that an eavesdropper who can obtain several million encrypted packets whose first byte of plaintext is known can deduce the base RC4 key by exploiting properties of the RC4 key schedule.

Within a week, Stubblefield, Ioannidis, and Rubin experimentally implemented the attack, and demonstrated that real systems could be cracked [10]. The first octet encrypted under WEP is a known fixed value, and they found that the required ciphertext packets can be readily obtained after eavesdropping on a network for a few hours.

Since then, others implemented the Fluhrer-Mantin-Shamir (FMS) attack and publicly released tools automating the process of breaking into WEP-protected networks. Because the attack uses off-the-shelf hardware and software, it is a serious threat. Also, because the attack is purely passive, detection is nearly impossible. Experiments in the field indicate that, with proper equipment, it is practical to eavesdrop on WEP-protected networks from distances of a mile or more from the target.

The FMS attack is devastating to WEP. Once the WEP key is discovered, all security is lost. The security risks are:

- The attacker can decrypt intercepted packets and read encrypted traffic, defeating the WEP confidentiality goals.
- The attacker can forge new encrypted packets that will be accepted by the access point, and join the wireless network, or attack other hosts, defeating the WEP integrity and authentication goals.

This situation puts all WEP-protected networks at serious risk for intrusion.

Fortunately, the 802.11 community received several months of advance warning of problems in WEP prior to the FMS attack: earlier, cryptographers had found other security problems in WEP. As a sampling, in the fall of 2000, Walker noted that the small IV size creates a serious risk of keystream reuse, a condition that allows an eavesdropper to recover plaintext traffic [11]. Then, in January 2001, Borisov, Goldberg, and Wagner showed several other attacks, including the fact that encrypted messages could be modified freely by an attacker without fear of detection, as well as the fact that the user authentication protocol is trivially defeated [2]. Arbaugh subsequently turned this into a practical attack that could decrypt any chosen packet in a few hours [1]. Other attacks against the authentication infrastructure and specific implementations have been uncovered, but they are outside the scope of this article. In summary, the problems with the design of WEP are as follows:

- 24-bit IVs are too short, and this puts confidentiality at risk.
- The CRC checksum, called the Integrity Check Value (ICV), used by WEP for integrity protection,

is insecure, and does not prevent adversarial modification of intercepted packets.
- WEP combines the IV with the key in a way that enables cryptanalytic attacks. As a result, passive eavesdroppers can learn the key after observing a few million encrypted packets.
- Integrity protection for source and destination addresses is not provided.

In each case, cryptographic algorithms were used inappropriately: the WEP designers selected well-regarded algorithms, such as RC4, but used them in insecure ways, often repeating well-documented mistakes. The lesson is that security protocol design is very difficult, and is best performed with an abundance of caution, supported by experienced cryptographers and security protocol designers.

TKIP: IEEE 802.11i Short-Term Solution

Most existing IEEE 802.11 systems implement WEP in hardware. To address the WEP vulnerabilities on this hardware, TGi has defined the Temporal Key Integrity Protocol, or TKIP. Installation will probably include a firmware upgrade and a driver upgrade. TKIP is intended only as an interim solution. The requirement to run on deployed hardware imposes several constraints:

- Allow deployed systems to be software or firmware upgradeable;
- Allow the current WEP hardware implementation to remain unchanged; and
- Minimize performance degradation imposed by the fixes.

TKIP is a set of algorithms that adapt the WEP protocol to address the known flaws while meeting these constraints. TKIP wraps WEP in three new elements:

- A message integrity code (MIC), called Michael, to defeat forgeries;[1]
- A packet sequencing discipline, to defeat replay attacks; and
- A per-packet key mixing function, to prevent FMS attacks.

TKIP mandates fresh keys—never-before-used, unrelated cryptographic keys—to address key reuse. The IEEE 802.1X key management scheme provides fresh keys.

Michael. A MIC algorithm calculates a keyed function of data at the transmitter, sends the resulting value as a tag with the data to the receiver, where it recomputes the value and compares the computed result with the tag accompanying the data. If the two tags match, the receiver accepts the data as authentic; if not, the receiver rejects the data as a forgery.

Example MICs include HMAC-SHA1, used by IPsec, and DES-CBC-MAC, widely used in financial applications. Conventional MICs are too computationally expensive to execute on existing hardware without unduly degrading performance. Therefore, TGi adopted a new MIC called Michael [4].

Michael uses a 64-bit key, and partitions packets into 32-bit blocks. Michael then uses shifts, exclusive ORs, and additions to process each 32-bit block into two 32-bit registers that will represent the final output, a 64-bit authentication tag. Michael limits the instruction set to minimize performance impact. It costs about 3.5 cycles/byte on a ARM7 processor and about 5.5 cycles/byte on a i486 processor. Some performance degradation is expected, but cheaper alternatives exhibiting appropriate security characteristics are unknown.

The security level of a MIC is measured in bits. If the security level of a MIC is s bits, then, by definition an attacker can on average construct a forgery in about 2^{s+1} packets. To meet its performance goals, Michael was designed to provide only about 20 bits of security. Michael is too weak to stand alone, so TKIP mandates countermeasures: TKIP requires a rekey after detecting a MIC validation error, and limits rekeying to once per minute. With this design the maximum expected number of false positives is about one per year.

Packet Sequencing. Replayed packets cannot be detected easily by a MIC alone. The usual way to address replay is to bind a packet sequence number to each packet with a MIC, which is used to enforce packet sequencing at the receiver, and reinitialize the sequence space whenever the MIC key is replaced.

TKIP extends the current WEP format to use a 48-bit sequence number, but, due to existing implementation constraints, associates the sequence number with the encryption key instead of the MIC key. TKIP mixes the sequence number into the encryption key (see the following section), and encrypts the MIC and the WEP ICV. This design translates replay attacks into ICV or MIC failures.

Per-Packet Key Mixing. As explained previously, concatenating the base key to the 24-bit WEP IV enables an attacker to recover the WEP encryption key via FMS attacks. To defend against them, TKIP introduces a new per-packet encryption key construction, based on a mixing function. The mixing function takes the base key, transmitter MAC address, and packet sequence number as inputs, and outputs a new per-packet WEP key. To minimize computational require-

[1]The literature refers to this concept as a Message Authentication Code (MAC). However, IEEE 802 had already appropriated the acronym MAC to designate Media Access Control.

ments, the mixing function is split into two phases.

The first phase uses a nonlinear substitution table, or S-box, to combine the base key, the transmitter MAC address, and the four most significant octets of the packet sequence number to produce an intermediate value. The intermediate value can be cached and used for up to 2^{16} packets. Since it includes the transmitter address, the mixing function produces a different value on each host, even when the same base key is used across hosts.

The second phase mixes the intermediate value with the two least significant octets of the packet sequence number, and produces a per-packet key. It uses a small cipher to diffuse the intermediate value and sequence number octets evenly throughout the per-packet key. The second phase decorrelates the packet sequence number from the per-packet key, thwarting FMS attacks; it costs about 150 cycles per packet.

There is no quantitative security analysis of the TKIP key mixing function. However, cryptographic review thus far suggests it achieves its design goals.

TKIP Keys. TKIP requires two distinct keys: a 128-bit key, used by the mixing function to produce a per-packet encryption key, and a 64-bit key, employed by Michael. TKIP assumes these keys are fresh at the start of each association. TGi has adopted IEEE 802.1X to provide both authentication and key management. IEEE 802.1X authenticates after association, then derives a fresh master key, and finally distributes this key for subsequent use. The station and the access point use the distributed master key to derive the pair of keys needed by TKIP. The details of this mechanism are beyond the scope of this article.

	WEP	TKIP	CCMP
Cipher Key Size(s)	RC4 40- or 104-bit encryption	RC4 128-bit encryption, 64-bit authentication	AES 128-bit
Key Lifetime Per-packet key	24-bit wrapping IV Concatenate IV to base key	48-bit IV TKIP mixing function	48-bit IV Not needed
Integrity Packet Header	None	Source and destination addresses protected by Michael	CCM
Packet Data Replay detection	CRC-32 None	Michael Enforce IV sequencing	CCM Enforce IV sequencing
Key Management	None	IEEE 802.1X	IEEE 802.1X

Comparison of security protocol features.

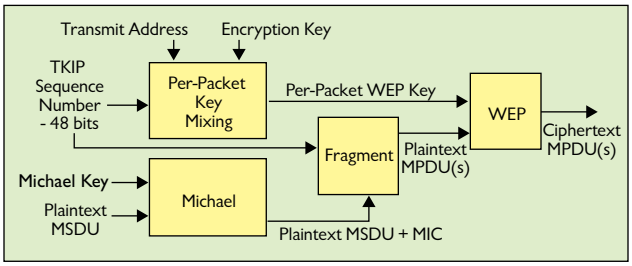

Figure 1. Flow of TKIP processing.

TKIP Summary. Figure 1 depicts TKIP as a front-end process to WEP. TKIP applies the per-packet key mixing function and WEP encryption to packet fragments (MPDUs), but the Michael MIC function applies to whole packets (MSDUs). No one would use this kind of design approach without implementation constraints.

The interrelationship of the TKIP components is believed to enhance its overall security. TKIP uses RC4 to encrypt the MIC, which decreases the information about the MIC key visible to an attacker. An attack that changes the packet sequence number also changes the per-packet encryption key, making it likely that both WEP ICV and TKIP MIC will decrypt incorrectly. Michael and its countermeasures make attacks that alter encrypted data computationally infeasible. Since the MPDU is protected from random bit-errors by both the IEEE 802.11 Frame Check Sequence (FCS) and by the WEP ICV, a valid FCS and ICV but invalid MIC implies the packet is most likely a forgery. Finally, since the MIC protects the source and destination addresses from change, packets can no longer be redirected to unauthorized destinations or the source spoofed.

CCMP: IEEE 802.11i Long-Term Solution

CCMP stands for the Counter-Mode-CBC-MAC Protocol. Like TKIP, the long-term solution addresses all known WEP deficiencies, but without the shackles of already-deployed hardware. The Advanced Encryption System (AES) [8] was selected for the encryption algorithm.

Mode of Operation. None of the existing AES modes of operation [3] offers a suitable balance of features required by this application. The following features are desirable:

- Use a single key to provide confidentiality and integrity, to reduce key management overhead and minimize the time spent computing AES key schedules.
- Provide integrity protection for the plaintext packet header, as well as integrity and confidentiality of the packet payload.
- Allow precomputation to reduce latency. Since packets can be lost, the receiver may perform precomputation for a packet that never arrives. However, the sender's efforts are rarely discarded.
- Support pipelining to increase throughput.
- Small implementation size, to keep costs reasonable.
- Small overhead for each packet.
- Avoid modes that are encumbered by patents (or pending patents).

A new mode called CCM was designed to meet all these criteria.

CCM merges two well-known and widely deployed techniques. CCM uses counter mode for encryption and the Cipher Block Chaining Message Authentication Code (CBC-MAC) [9] for integrity protection. Both algorithms employ only the encryption primitive at both the sender and the receiver. CCM has been submitted to NIST for consideration as a Federal Information Processing Standard (seecsrc.nist.gov/encryption/ modes/proposedmodes/).

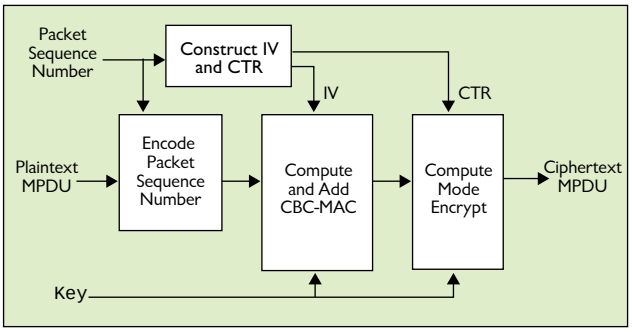

Figure 2. CCM Mode provides confidentiality and integrity.

CCM uses the same key for both confidentiality and integrity. This is normally a dangerous practice, but CCM avoids the pitfalls of this usage by guaranteeing that the space for the counter mode never overlaps with that used by the CBC-MAC initialization vector. The intuition behind CCM mode is that if AES behaves like a pseudo-random permutation, then the output of the cipher operating on each of these two spaces will be independent.

The CCMP Protocol. The protocol using CCM has many properties in common with TKIP. Freedom from constraints associated with current hardware leads to a more elegant solution. Figure 2 illustrates the use of CCM for a single WLAN packet fragment (MPDU). As with TKIP, CCMP employs a 48-bit IV, ensuring the lifetime of the AES key is longer than any possible association. In this way, key management can be confined to the beginning of an association and ignored for its lifetime. CCMP uses a 48-bit IV as a sequence number to provide replay detection, just like TKIP.

AES has significantly different properties from the RC4 encryption algorithm used by WEP and TKIP. AES obviates any need for per-packet keys, so CCMP has no per-packet key derivation function. CCMP uses the same AES key (and associated AES key schedule) to provide confidentiality and integrity protection for all of the packets in an association. The CCM MIC length is adjustable between two octets and sixteen octets. CCMP uses an 8-octet MIC, which is significantly stronger than Michael. Unlike TKIP and WEP, the encrypted ICV is no longer required.

TKIP provides integrity protection over the whole MSDU and confidentiality over MSDU, leading to implementation complexity. Since CCM provides both services, it is straightforward to provide confidentiality and integrity protection over the same data structure. CCMP must, however, protect nearly the entire packet header to defend against fragmentation attacks.

Summary

We've reviewed here the problems with WEP and briefly described the new protocols designed to replace it. To summarize the discussion, the table appearing here compares the features of WEP, TKIP, and CCMP. WEP meets none of its security goals because of misuse of cryptographic primitives. The new protocols, TKIP and CCMP, address all known WEP problems. As they are deployed, it is expected the new protocols will finally cause the struggle between hacker and defender to shift to layers above the wireless MAC. C

References

1. Arbaugh, W. *An Inductive Chosen Plaintext Attack Against WEP/WEP2.* IEEE Document 802.11-02/230, May 2001; grouper.ieee.org/groups/802/11.
2. Borisov, N., Goldberg, I., and Wagner, D. Intercepting mobile communications: The insecurity of 802.11. In *Proceedings of the International Conference on Mobile Computing and Networking,* (July 2001), 180–189.
3. Dworkin, M. *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*. NIST Special Publication 800-38A, Dec. 2001.
4. Ferguson, N. *Michael: An improved MIC for 802.11 WEP.* IEEE 802.11 doc 02-020r0, Jan. 17, 2002; grouper.ieee.org/groups/802/11.
5. Fluhrer, S., Mantin, I., and Shamir, A. Weaknesses in the key schedule algorithm of RC4. In *Proceedings of the 4th Annual Workshop on Selected Areas of Cryptography*, 2001.
6. Housley, R. and Arbaugh, W. Security problems in 802.11-based networks. *Commun. ACM 46*, 5 (May 2003).
7. ISO/IEC 8802-11 ANSI/IEEE Std 802.11-1999. Information technology—Telecommunications and information systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) specifications.
8. National Institute of Standards and Technology. FIPS Pub 197: *Advanced Encryption Standard (AES)*. Nov. 26, 2001.
9. National Institute of Standards and Technology. FIPS Pub 113: *Computer Data Authentication*. May 30, 1985.
10. Stubblefield, A., Ioannidis, J., and Rubin, A. Using the Fluhrer, Mantin, and Shamir attack to break WEP. In *Proceedings of the 2002 Network and Distributed Systems Security Symposium* (2002), 17–22.
11. Walker, J. Unsafe at Any Key Size: An Analysis of the WEP Encapsulation. IEEE 802.11 doc 00-362, Oct. 27, 2000; grouper.ieee.org/groups/802/11.

Nancy Cam-Winget (ncamwing@cisco.com) is a technical leader at Cisco Systems in San Jose, CA.
Russ Housley (housley@vigilsec.com) is the founder of Vigil Security in Herndon, VA.
David Wagner (daw@cs.ucberkeley.edu) is an assistant professor at the University of California at Berkeley.
Jesse Walker (jesse.walker@intel.com) is a security architect at Intel Corporation in Hillsboro, OR, and editor for the IEEE 802.11i security enhancements.