Security Considerations for IEEE 802.15.4 Networks

Naveen Sastry
University of California, Berkeley
nks@cs.berkeley.edu

David Wagner
University of California, Berkeley
daw@cs.berkeley.edu

ABSTRACT

The IEEE 802.15.4 specification outlines a new class of wireless radios and protocols targeted at low power devices, personal area networks, and sensor nodes. The specification includes a number of security provisions and options. In this paper, we highlight places where application designers and radio designers should exercise care when implementing and using 802.15.4 devices. Specifically, some of the 802.15.4 optional features actually reduce security, so we urge implementors to ignore those extensions. We highlight difficulties in safely using the security API and provide recommendations on how to change the specification to make it less likely that people will deploy devices with poor security configurations.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection Access controls; D.4.6 [**Operating Systems**]: Security and Protection Authentication

General Terms

Security

Keywords

IEEE 802.15.4, Link Layer Security, Sensor Networks

1. INTRODUCTION

The growing importance of small and cheap wireless devices demands a common platform so that the devices can communicate with each other and share components to lower costs. The 802.15.4 specification [6] (802.15.4) describes wireless and media access protocols for personal area networking devices. The sensor network community has begun using these protocols as well. The protocols are intended for hardware implementation on a dedicated radio chip. The range of envisioned applications is broad, spanning wireless game controllers, environmental, medical, and building monitoring instruments, to heating and ventilation sensors [3, 14, 16, 17, 18]. These applications frequently use embedded devices controlled by an 8 or 16 bit microcontroller meant to operate without human intervention for months. The long period of unattended operation with underpowered hardware has two implications for the design of such systems: the software that runs on these devices must be simple and correct, and the devices must make efficient use of their limited energy. The wireless communication chip that is a part of these devices must honor these requirements, since it and the microcontroller represent the two largest sources of energy consumption.

The 802.15.4 specification is meant to support a variety of applications, many of which are security sensitive. For example, consider the case of a sensor network that measures building occupancy as an alarm system: there is an obvious privacy concern about tracking the people in the building. Additionally, if the network is not secured, an adversary could modify and inject messages to either cause an alarm, or more worryingly, to suppress legitimate alarm signals. Many applications require confidentiality and most have a need for integrity protection. The 802.15.4 specification addresses these needs through a link-layer security package.

There are two intended uses of the 802.15.4 API, one for application designers that will use the features of the wireless radio chip directly and one for higher level libraries and specifications that use the 802.15.4 API to export their own higher-level services. The ZigBee specification, still in development [1], is one important example of the latter.

Contributions. In this paper, we analyze the 802.15.4 protocol from a security standpoint and outline a number of problems in the specification. We can broadly classify the vulnerabilities we have observed in the 802.15.4 protocol as stemming from three distinct problems: IV management, key management, and insufficient integrity protection. The 802.15.4 specification mandates a particular key and IV format that leads to vulnerabilities. Additionally, we believe the specification committee did not include enough support for integrity. We propose several changes to the specification that we believe will improve security for 802.15.4 users.

2. 802.15.4 SECURITY OVERVIEW

A link layer security protocol provides four basic security services: *access control*, *message integrity*, *message confidentiality*, and *replay protection*.

WiSE'04, October 1, 2004, Philadelphia, Pennsylvania, USA.

Access control and message integrity. Access control means the link layer protocol should prevent unauthorized parties from participating in the network. Legitimate nodes should be able to detect messages from unauthorized nodes and reject them. Also, a secure network should provide message integrity protection: if an adversary modifies a message from an authorized sender while the message is in transit, the receiver should be able to detect this tampering. Including a *message authentication code* (MAC)[1] with each packet provides message authentication and integrity. A MAC can be viewed as a cryptographically secure checksum of a message. Computing it requires authorized senders and receivers to share a secret cryptographic key, and this key is part of the input to the computation. The sender computes the MAC over the packet with the secret key and includes the MAC with the packet. A receiver sharing the same secret key recomputes the MAC and compares it with the MAC in the packet. The receiver accepts the packet if they are equal, and rejects it otherwise. Message authentication codes must be hard to forge without the secret key. Consequently, if an adversary alters a valid message or injects a bogus message, she will not be able to compute the corresponding MAC, and authorized receivers will reject these forged messages.

Confidentiality. Confidentiality means keeping information secret from unauthorized parties. It is typically achieved with encryption. Preferably, an encryption scheme should not only prevent message recovery, but also prevent adversaries from learning even partial information about the messages that have been encrypted. This stronger property is known as *semantic security* [10].

One implication of semantic security is that encrypting the same plaintext two times should give two different ciphertexts. If the encryption process is identical for two invocations on the same message, then semantic security is clearly violated: the resulting ciphertexts are identical. A common technique for achieving semantic security is to use a unique *nonce* for each invocation of the encryption algorithm. A nonce can be thought of as a side input to the encryption algorithm. The main purpose of a nonce is to add variation to the encryption process when there is little variation in the set of messages. Since the receiver must use the nonce to decrypt messages, the security of most encryption schemes do not rely on nonces being secret. Nonces are typically sent in the clear and are included in the same packet with the encrypted data.

Replay Protection. An adversary that eavesdrops on a legitimate message sent between two authorized nodes and replays it at some later time engages in a *replay attack*. Since the message originated from an authorized sender it will have a valid MAC, so the receiver will accept it again. Replay protection prevents these types of attacks. The sender typically assigns a monotonically increasing sequence number to each packet and the receiver rejects packets with smaller sequence numbers than it has already seen.

2.1 Protocol Description

In this section, we will provide a brief outline of the 802.15.4 security architecture. We will provide more detail as needed when we outline more specific problems.

802.15.4. Addressing in 802.15.4 is accomplished via a 64-bit node identifier and a 16-bit network identifier. 802.15.4 supports a few different addressing modes. For example, a 16-bit truncated address may be used in place of the full 64-bit node identifier in certain cases. This allows the sizes of the source and destination addresses to vary between 0 and 10 bytes depending on whether truncated or full addresses are used, and whether or not the node sends to the broadcast address. For our purposes, it is sufficient to only consider the 64-bit node identifier as the address. We urge the interested reader to see the specification for complete details.

There are two important packet types that are relevant to the security of 802.15.4: data packets and acknowledgment packets. A data packet, seen in Figure 1(a), has variable length and is used by a node to send a message to a single node or to broadcast a message to multiple nodes. Each data packet has a flags field that indicates the packet type, whether security is enabled or not, the addressing modes that are in use, and whether the sender requests an acknowledgment. A 1 byte sequence number serves to identify the packet number for acknowledgments. The packet optionally includes source and destination addresses. As noted above, each field is variably sized between 0 and 10 bytes. The data payload field comes after the addressing fields. It is less than 102 bytes. Finally, a 2 byte CRC checksum field protects the packet against transmission errors.

The acknowledgment packet, as depicted in Figure 1(b), is sent by the recipient only if the corresponding data packet was not sent to a broadcast address and the sender requested an acknowledgment. Its format is simple: a 2 byte flags field similar to the one in the data packet, the 1 byte sequence number from the packet that it is acknowledging, and a 2 byte CRC. There is no addressing information in the acknowledgment packet.

Security. The 802.15.4 security layer is handled at the media access control layer, below application control. The application specifies its security requirements by setting the appropriate control parameters into the radio stack. If an application does not set any parameters, then security is not enabled by default. An application must explicitly enable security, as we detail below. The specification defines four packet types: beacon packets, data packets, acknowledgments packets, and control packets for the media access control layer. The specification does not support security for acknowledgement packets; other packet types can optionally support integrity protection and confidentiality protection for the packet's data field.

An application has a choice of *security suites* that control the type of security protection that is provided for the transmitted data. Each security suite offers a different set of security properties and guarantees, and ultimately different packet formats. The 802.15.4 specification defines eight different security suites, outlined in Table 1. We can broadly classify the suites by the properties that they offer: no security, encryption only (AES-CTR), authentication only (AES-CBC-MAC), and encryption and authentication (AES-CCM). Each category that supports authentication comes in three variants depending on the size of the

[1] The 802.15.4 specification refers to the message authentication code as a message integrity code (MIC) to differentiate it from media access control. In this paper, we follow the cryptographic convention of referring to the integrity code as a MAC, and we do not abbreviate media access control.

1 byte	2 bytes	1 byte	0/2/4/10 bytes	0/2/4/10 bytes	variable	2 bytes
Len.	Flags	Seq. No	Dest. Address	Source Address	Data payload	CRC

(a) Data packet format

1 byte	2 bytes	1 byte	2 bytes
Len.	Flags	Seq. No	CRC

(b) Acknowledgment packet format

Figure 1: Data and acknowledgment packet formats

4 bytes	1 byte	variable
Frame Counter	Key Ctr	Encrypted Payload

(a) AES-CTR

variable	4/8/16 bytes
Payload	MAC

(b) AES-CBC-MAC-b, $b \in \{4, 8, 16\}$ MAC size

4 bytes	1 byte	variable	4/8/16 bytes
Frame Counter	Key Ctr	Encrypted Payload	Encrypted MAC

(c) AES-CCM-b, $b \in \{4, 8, 16\}$ MAC size

Figure 2: The formatting of the data field (see Figure 1(a)) for the three main security suites. When included, the MAC protects the packet headers as well as data contents.

Name	Description
Null	No security
AES-CTR	Encryption only, CTR Mode
AES-CBC-MAC-128 AES-CBC-MAC-64 AES-CBC-MAC-32	128 bit MAC 64 bit MAC 32 bit MAC
AES-CCM-128 AES-CCM-64 AES-CCM-32	Encryption & 128 bit MAC Encryption & 64 bit MAC Encryption & 32 bit MAC

Table 1: Security suites supported by 802.15.4 (see Table 75 from the 802.15.4 specification). The specification mandates that implementations must support the AES-CCM-64 and Null suites while the other suites are optional (§7.6).

Address	Security Suite	Key	Last IV	Replay Ctr

Figure 3: Format of an ACL entry. The destination address of an outgoing packet is matched with the address field in an ACL entry. The packet is then processed using the indicated security suite with the key and IV listed in the ACL entry. For incoming packets, the source address is matched with the address field in the ACL entry. Cryptographic operations use the key from the ACL entry and the replay counter field acts as a high water mark if replay detection is enabled.

MAC that it offers. Each variant is considered a different security suite and has its own name. The MAC can be either 4, 8, or 16 bytes long. The longer the MAC the lower the chance that an adversary has to blind forgery by guessing an appropriate code. For example, with an 8 byte MAC, an adversary has a 2^{-64} chance of forging the MAC. The tradeoff is a larger packet size for increased protection against authenticity attacks. Additionally, for each suite that offers encryption, the recipient can optionally enable replay protection. Radio designers do not have to implement all of the suites. The specification only requires that radio chips provide support for the Null suite and the AES-CCM-64 suite.

An application indicates its choice of security suites based on source and destination addresses. 802.15.4 radio chips have an *access control list* (ACL) that controls what security suite and keying information to use. Compliant devices may support up to 255 ACL entries. Each entry contains an 802.15.4 address, a security suite identifier, and *security material*, as shown in Figure 3. The security material is persistent state necessary to execute the security suite. It consists of the cryptographic key and, for suites that provide encryption, the nonce state that must be preserved across different packet encryption invocations. When replay protection is invoked, the security material also includes a high water mark of the most recently received packet's identifier.

As a part of the interface for sending packets, the application must specify a boolean indicating whether security is enabled. If no security is requested, the packet is sent out as is. If security is enabled, the media access control layer looks up the destination address in its ACL table. If there is a match ACL entry, the security suite, key, and nonce specified in that ACL entry are used to encrypt and/or authenticate the outgoing packet, and the flags field on outgoing packet is set accordingly. If the destination address is not listed in the ACL table, a *default ACL entry* is used instead; the default ACL entry is similar to the other ACL entries except that it matches all destination addresses. If the default ACL entry is empty and the application has requested security, the media access control layer returns an error code.

On packet reception, the media access control layer consults the flags field in the packet to determine if any security suites have been applied to that packet. If no security was used, the packet is passed as is to the application. Otherwise, the media access control layer uses a similar process to find the appropriate ACL entry, this time based on the sender's address. It then applies the appropriate security suite, key, and replay counter to the incoming packet, presenting the application with an error message if no appropriate ACL entry could be located.

We will now provide more detail about the categories of security suites:

- **Null:** This is the simplest security suite. Its inclusion is mandatory in all radio chips. It does not have any security material and operates as the identity function. It does not provide any security guarantees.

- **AES-CTR:** This suite provides confidentiality protection using the AES block cipher [15] with counter mode. To encrypt data under counter mode, the sender breaks the cleartext packet into 16-byte blocks $p_1, \ldots, p_n$ and computes $c_i = p_i \oplus E_k(x_i)$. Each 16-byte block uses its own varying counter, which we call x_1. The recipient recovers the original plaintext by computing $p_i = c_i \oplus E_k(x_i)$. Clearly, the recipient needs the counter value x_i in order to reconstruct p_i. The x_i counter, known as a *nonce* or *IV*, is composed of a static flags field, the sender's address, and 3 separate counters: a 4 byte frame counter that identifies the packet, a 1 byte key counter field, and a 2 byte block counter that numbers the 16 byte blocks within the packet as shown in Figure 4. The frame counter is maintained by the hardware radio. The sender increments it after encrypting each packet. When it reaches its maximum value, the radio returns an error code and no further encryptions are possible. The key counter is a one byte counter under the application's control. It can be incremented if the frame counter ever reaches its maximum value. The requirement is that the nonce must never repeat within the lifetime of any single key, and the role of the frame and key counters is to prevent nonce reuse. The 2 byte block counter ensures that each block will use a different nonce value; the sender does not need to include it with the packet, since the receiver can infer its value for each block.

  In summary, the sender includes the frame counter, key counter, and encrypted payload into the data payload field of the packet, as in Figure 2(a).

- **AES-CBC-MAC:** This suite provides integrity protection using CBC-MAC [11]. The sender can compute either a 4, 8, or 16 byte MAC using the CBC-MAC algorithm, leading to three different AES-CBC-MAC variants. The MAC can only be computed by parties with the symmetric key. The MAC protects packet

1 byte	8 bytes	4 bytes	1 byte	2 bytes
Flags	Source address	Frame Ctr	Key Ctr	Block Ctr

Figure 4: The format of the input (x_i) to the block cipher for the AES-CTR and AES-CCM suites. The flags field is a constant for AES-CTR mode; its value is prescribed by the CCM specification for AES-CCM.

headers as well as the data payload. The sender appends the plaintext data with the MAC, as in Figure 2(b). The recipient verifies the MAC by computing the MAC and comparing it with the value included in the packet.

- **AES-CCM:** This security suite uses CCM mode for encryption and authentication [19]. Broadly, it first applies integrity protection over the header and data payload using CBC-MAC and then encrypts the data payload and MAC using AES-CTR mode. As such, AES-CCM includes the fields from both the authentication and encryption operations: a MAC, and the frame and key counters. These fields serve the same function as above. Just as AES-CBC-MAC has three variants depending on the MAC size, AES-CCM also has three variants. We show its packet format in Figure 2(c).

A receiver can optionally enable replay protection when using a security suite that provides confidentiality protection. This includes AES-CTR and all of the AES-CCM variants. The recipients use the frame and key counter as a 5 byte value, the *replay counter*, with the key counter occupying the most significant byte of this value. The recipient compares the replay counter from the incoming packet to the highest value seen, as stored in the ACL entry. If the incoming packet has a larger replay counter than the stored one, then the packet is accepted and the new replay counter is saved. If, however, the incoming packet has a smaller value, the packet is rejected and the application is notified of the rejection. We refer to this counter as the replay counter, even though it is the same counter as the nonce. It serves a logically different purpose from the nonce, which is used for confidentiality. The replay counter is not exposed to the application to use.

2.2 Keying Models

Symmetric cryptography relies on both endpoints using the same key when communicating securely. In a group of nodes, the *keying model* governs what key a node uses to communicate with another node. The keying model that is most appropriate for an application depends on the threat model that an application faces and what types of resources it is willing to expend for key management. For example, in the network shared key model, every node uses the same key for communicating with every other node. Each node only needs to keep track of a single key, which eases the management problems. In this section, we define a set of common keying models. In Section 3, we discuss the problems with the 802.15.4 architecture in supporting these different keying models.

We present a few of the more common keying models that are appropriate for sensor networks:

- **Network shared keying:** With a single network-wide shared key, each node in the system possesses the same key and uses it to communicate with all other nodes. Key management becomes trivial with this approach since all communication uses the same key. Additionally, the memory requirements are minimal. Applications can therefore use the network shared key with little effort.

  However, the management simplicity comes at the cost of a vulnerability to insider attacks. It is more vulnerable than other keying models to a single key compromise, as happens when an adversary compromises a single node. An adversary can use the compromised node to undermine the security guarantees of the entire network. The single node can break the confidentiality of any message sent in the system and forge messages claiming to originate from any node. If we expect nodes to be occasionally compromised or captured, network shared keying will be less attractive.

- **Pairwise keying:** Pairwise keying tolerates node compromise by limiting the scope of every key. With pairwise keying, each pair of nodes share a different key. Thus, a node compromise only affects past and future messages sent to or from that node; other traffic is unaffected. This provides better security than network shared keying.

  The greater robustness against node compromise does come at a cost, particularly in the overhead for key management. If a node communicates with many other nodes, it must store many keys and select the appropriate one when communicating. On devices with minimal resources, the storage costs can be prohibitive.

- **Group keying:** Group keys are a compromise between network shared keys and pairwise keys. A single key is shared among a set of nodes and is used on all links between any two nodes in that group. The partition into groups may be made based on location, network topology, or similarity of function. The advantage of group keying is that it provides an intermediate tradeoff between network shared keying and pairwise keying, with partial resistance to node compromise at a lower cost than pairwise keying.

- **Hybrid approaches:** Some systems may use a combination of the above keying models simultaneously in the same application. For example, we might use pairwise keying for all links between a node and a base station and use a network shared key for all other links.

2.3 Implementations

We are not aware of any wireless chip that fully supports the security operations detailed in the 802.15.4 standard in hardware. The Atmel Z-Link transceiver [7] and Motorola

MC13192 [9] support the 802.15.4 PHY standard that governs radio packet formats and frequencies, but not the entire media access control portion of the standard. One would have to implement the security operations on the microcontroller in order to interact with 802.15.4 devices that use security. The Chipcon CC2420 does have hardware support for the cryptographic primitives that allows for seamless operation with 802.15.4 devices that follow the specification. However, as we detail in Section 3.2.3, they do not adhere to the interface outlined in the 802.15.4 specification, so they do not support multiple keys well.

We believe that hardware implementations of the standard are valuable in order to simplify the application writer's job. Handling the cryptographic operations in hardware frees the microcontroller from the real time demands required to decrypt and encrypt the packets.

3. PROBLEMS

We have found several vulnerabilities and pitfalls in 802.15.4. They fall into three categories: IV management, key management, and integrity protection. Each represents danger zones for application developers, where it is easy to use 802.15.4 in a way that provides less security than one might expect. In some cases, there are easy workarounds that developers can employ today; others can only be fixed with substantial changes to the specification.

3.1 IV Management Problems

3.1.1 Same Key in Multiple ACL Entries

As noted, there are up to 255 ACL entries used to store different keys and their associated nonce. The sender chooses the appropriate ACL entry based on the destination address.

However, there is a vulnerability if the same key is used in two different ACL entries. In that case, it is highly likely that the sender will accidentally reuse the nonce. For example, suppose a sender uses the AES-CCM-64 security suite with the same key k for recipient r_1 and recipient r_2 and initializes the frame and key counters to `0x0` for both recipients. If the sender transmits message m_1 with data `0xAA00` to r_1 and then message m_2 with data `0x00BB` to r_2, the sender will end up reusing the same nonce (frame counter of `0x0` and key counter of `0x0`). This is because each recipient has their own ACL entry with independent nonce state. Because AES-CCM uses CTR mode, which acts like a stream cipher, an adversary can easily recover the `xor` of the plaintexts by computing the `xor` of the two ciphertexts, in this case `0xAABB`, completely breaking the confidentiality property[2].

There are a number of ways that two separate recipients might end up with the same key in two separate ACL entries:

- Coarse grained ACL control. As we note in Section 3.2.1, group keys are not well supported. An application designer may be tempted to implement group keying by creating two separate ACL entries sharing the same key. This will appear to work but it will not be secure.
- Race conditions in routing change protocols. Suppose that a node uses key k_1 to secure communications to any parent node in the routing tree and key k_2 to secure all other communications. Key k_2 will be listed as the default ACL entry, with a separate ACL entry specifying address p and key k_1. If the application decides to switch parents from p to p' by adding an ACL entry for p' before removing p's entry, there is a race condition where the same key is active in the ACL list at the same time.

We believe nonce reuse will likely occur if an application ever sets up the same key in two different ACL entries. In such cases, confidentiality will be violated, though integrity is unaffected. The problem is that this error can occur all too easily if application programmers are not vigilant.

We should point out that a sender *can* safely send two messages to two different recipients using the same key as long as it carefully manages the nonce state. The easiest method is to use a single ACL entry. The sender can then send the first message, change the destination in the ACL entry, and then send the second message. The general principle to prevent nonce reuse is that the nonce state should never be separated from the key.

3.1.2 Loss of ACL State Due to Power Interruptions

We expect that many 802.15.4 devices will be battery or solar powered. Radio chip designers must ensure that the ACL state is properly maintained even during power interruptions and low-power operation.

Power Failure. Consider what happens if the ACL state is lost when the node encounters a power failure. If no special precautions are taken, the node will emerge with a cleared ACL table when power is restored. Presumably, the node's software can then repopulate the ACL table with the appropriate keys. However, it is not clear what to do about the nonce states. If all nonces are reset to a known value, such as 0, nonces will be reused, compromising security.

Application designers can avoid nonce reuse in a number of ways if they can detect the power disruption. Firstly, nodes can establish new keys after a power disruption, so that they don't reuse the same nonce twice with the same key. Alternatively, they can always store the key counter in indelible flash memory, incrementing it after sending each packet. However, since storing values in flash memory is slow and energy inefficient, application designers can amortize the cost of writing to the flash by "leasing" a block of counter values and storing the lease in flash memory. They can acquire a new block of counter values when they run out of values and store the lease information in the flash memory. After a reboot, they can invalidate all counter values up to or including the lease currently in flash and acquire the next block of values.

This is a dangerous pitfall: applications not designed with power failures in mind can easily end up with a product that appears to work but actually fails to secure communications against eavesdroppers. Application designers must take this into account in order to design nodes that are secure across power interruptions.

Low Powered Operation. A related problem to power failure is how to preserve the nonce state if the node goes into low powered operation. In order to extend the battery's lifetime, the devices must engage in some kind of duty cycling

[2] Recall that the ciphertext for a message is $m \oplus E_k(x_1)$. So when the same counter is used for both messages, the `xor` of the ciphertexts will be $[m_1 \oplus E_k(x_1)] \oplus [m_2 \oplus E_k(x_1)] = m_1 \oplus m_2$, where x_1 is the nonce.

whereby parts of the device are on for only a small fraction of the time. Since an 802.15.4 radio can consume 19.7 mA at 1.8V [8] just to receive an incoming packet, keeping the radio chip in a lower power mode can greatly increase the device's power consumption efficiency. If it emerges from its low-powered state with a cleared ACL, it will again reuse nonce values and break confidentiality. Maintaining the nonce through low powered operation is easier to solve since the radio chip knows when it will enter and leave low-powered mode; since this can occur frequently, the radio chip must use efficient mechanisms. Unfortunately, the specification does not address the issue of what happens to the wireless chip in a low-power state.

Saving and restoring nonce state in software is a reasonable fix. However, this solution is not cheap: each ACL entry has at least 10 bytes of state that needs to be stored (5 bytes for the inbound replay counter and 5 bytes for the outbound nonce that it uses for that key). With 255 ACL entries, this is more than 2 kilobytes of state, imposing significant memory and computational overheads. Another alternative, maintaining power to the ACL's RAM so that the the ACL contents is not lost during low-powered operation is workable; the downside is an increased power consumption in low powered operation.

3.2 Key Management Problems

The second class of problems results from inadequate support in the ACL table for many keying models.

3.2.1 No Support for Group Keying

Supporting group keying under 802.15.4 is unwieldy. For example, suppose that nodes $n_1, \ldots, n_5$ wish to communicate amongst themselves using key k_1, while nodes $n_6, \ldots n_9$ use key k_2. Because each ACL entry can only be associated to a single destination address (§7.5.8.1), there is no good way to support this desired model.

One tempting approach is to create five ACL entries, one for each of nodes $n_1, \ldots, n_5$, all mentioning the same key k_1. This requires that the 802.15.4 radio's ACL table be large enough to hold all these entries. More seriously, as we saw in Section 3.1.1, this approach poses unacceptable security risks: because each entry stores a nonce separately, we are likely to encounter nonce reuse, exposing plaintext. Consequently, we discard this approach as simply too dangerous.

Another tempting approach is to create a single ACL entry for key k_1. Before sending to node n_1, the destination address associated with that ACL entry could be changed to mention n_1. If the application later wants to send a message to n_2 using the same key, it must switch the destination address associated with that the ACL entry. In general, the destination address must be modified every time a packet is sent. This makes packet transmission cumbersome. The real problem, though, is on the receiving side. If the receiver performs the same address switching trick, the receiver will need to ensure it has an ACL entry for the sender *before* the message from that sender ever arrives. The receiver therefore must always be able to predict which node from the group will next send it a message, so that the ACL entry can be set up appropriately. Except in special cases (such as restricted communication patterns), this is likely to pose too many constraints to be workable.

In short, there appears to be no simple way to use group keying securely in 802.15.4 networks. We hypothesize that many, if not all, of those who naively attempt to use group keying in 802.15.4 will end up with configurations that appear to be functional but actually compromise security. This is a shame, for group keying is very natural for many problems.

3.2.2 Network Shared Keying Incompatible with Replay Protection

When using a single network-wide shared key, there is no way to protect against replay attacks. To use the network shared key model, an application must use the default ACL entry, since as we saw in Section 3.2.1, the other ACL entries are not useful for group communication. Recall that the default ACL entry will be used when there is no matching ACL entry.

Now, suppose that the network shared key is loaded into the default ACL and node s_1 sends 100 messages using replay counters $0 \ldots 99$. The recipient gets these packets and would like to perform replay protection. It must keep a high water mark of the largest replay counters it has seen so far. According to the specification (§7.6.2.3.2 and §7.6.3.3.2), the receiver updates the replay counter associated with the default ACL as each packet arrives. Now, if sender s_2 sends a message with its replay counter starting at 0, the recipient will reject the message since it will only accept messages with replay counters greater than 99. Therefore, for the recipient to use replay protection with the shared key model, the senders must coordinate their use of replay counter space. This is not feasible when there are more than a handful of members in a group, precluding the use of replay protection with the shared network key.

3.2.3 Pairwise Keying Inadequately Supported

We also note that the specification could include stronger support for pairwise communication. The specification allows a 802.15.4 radio to have up to 255 ACL entries, but it does not specify a required minimum number of ACL entries. In particular, compliant radio chips need not have more than one or two ACL entries. As an example, the Chipcon CC2420, has support for only two keys[3] [8]. As detailed in Section 3.2.1, an ACL entry cannot be safely shared among a group of multiple nodes. This means that in a pairwise keying model, a radio chip with support for n ACL entries will limit us to networks containing at most about n nodes. This poses a significant limit to scalability, and it means that pairwise keying will only be feasible on radio chips with support for a large number of ACL entries. To support pairwise keying, we submit that it may make sense to revise the specification to mandate a reasonable minimum number of ACL entries.

3.2.4 Discussion

The source of these difficulties is partly a result of confusing the role of a nonce and a replay counter. The nonce sent in outgoing packets serves two purposes: it provides a non-repeating value that protects confidentiality; and, it provides

[3]Instead of a default ACL entry and ACL list, the CC2420 actually has two ACL entries, named key 0 and key 1. The ACL entries do not have an address associated with them, as the specification requires. Instead, there are two registers specifying which key is to be used for transmission and reception, respectively. They have similar properties to the default ACL, so the CC2420 can only effectively support one group at a time. It does not support pairwise keying well.

a monotonically increasing counter that prevents replay attacks. To protect confidentiality, the sender must ensure that it never uses the same nonce twice for the same key. To protect against replay attacks, the recipient needs to ensure that every sender uses a larger nonce value than found in previous messages sharing the same key. The first requirement suggests that the nonce should be strongly bound to the key: anytime the key is used for encryption, it must use a different nonce value. Therefore, any uses of the key, even in different ACL entries, should share a single nonce register. Conversely, the second requirement suggests that the replay counter should be strongly bound to the sender's address. If the same key appears in multiple ACL entries, we should maintain a separate high-water mark for each sender.

The current ACL structure does not support these two requirements, leading to the problems in properly supporting replay when used with a group keying model. The problem stems from overloading the nonce to serve both as an IV and as a replay counter with the ACL structure they mandate. The opposing requirements for sending to ensure confidentiality and receiving to guarantee replay protection place different demands on the ACL structure. The specification implicitly assumes that a key does not appear in more than one ACL entry at a time, otherwise it will encounter problems sending as described in Section 3.1.1. Furthermore, the specification indicates that the recipient maintains a single nonce for each *key value* used for outbound communication. It also only has a single inbound slot to store the replay counter, shared among all senders using that key. Hence, when the recipient receives packets from different senders, as occurs with group keying or a network keying, the recipient cannot update the replay counter properly, making replay protection incompatible with network shared and group keying.

We conclude that none of the three most important keying models are well supported by 802.15.4. Each has some shortcoming:

- **Network shared keying** is incompatible with replay protection.
- **Pairwise keying** is only useful if the radio chip designer includes a sufficient number of ACL entries. This is not required by the specification, and one existing 802.15.4 radio is poorly suited for pairwise keying.
- **Group keying** is not workable, as described in Section 3.2.1.

These problems will not affect every application, but they do restrict an application's choice of keying models and may force undesirable compromises. Pairwise keying may be workable if the radio provides enough ACL entries, but otherwise it cannot be used. Network shared keying can only be used if the application does not require replay protection.

3.3 Insufficient Integrity Protection

3.3.1 Unauthenticated Encryption Modes

As detailed in Table 1, the AES-CTR suite uses counter mode without a MAC (see §7.6.2). The standard does not mandate that radio designers support the CTR mode suite; only AES-CCM-64 is mandatory. However, we believe that AES-CTR is so dangerous that it should never be enabled or implemented. Unauthenticated encryption introduces a significant risk of protocol level vulnerabilities.

Researchers have found a number of vulnerabilities in protocols that use encryption protected only by a CRC and not a cryptographically strong message authentication code. All of the attacks center on the fact that in the course of modifying the ciphertext, the adversary can construct appropriate modifications to the CRC so that the receiver accepts the packet. Researchers have discovered unauthenticated encryption vulnerabilities in IPSec [12], 802.11 [13],and SSH version 1 [2, 4, 5] that compromise not only integrity but also confidentiality. Any application that uses AES-CTR security suite becomes vulnerable to similar attacks. As Bellovin noted, developers have a tendency to assume that "since decrypting with the wrong key will yield garbage, additional integrity checking is not needed" [12], but this assumption is simply inaccurate.

What is perhaps surprising to those not trained in cryptography is that failures of integrity can affect confidentiality as well. The root of the problem is that, in many systems, the ability to forge unauthentic messages often allows an attacker to trick an endpoint into disclosing secrets. The severity of this problem depends on the details of the specific application protocol, so it is impossible to make any definitive statements that will apply to all deployments. Nonetheless, time has proven again and again that use of encryption without a MAC poses a significant risk of security breaches. Therefore, we recommend that AES-CTR should never be used by application designers, should not be supported by 802.15.4 devices, and should be stricken from the standard.

3.3.2 Denial-of-service Attacks on AES-CTR

Next, we show a simple, single-packet denial-of-service attack that is applicable when a 802.15.4 network uses the AES-CTR suite with replay protection enabled. Suppose a sender s and recipient communicate with the AES-CTR suite using key k, and the recipient has enabled replay protection. Recall that the recipient maintains a high water mark composed of the key and frame counter, rejecting packets whose counter is smaller than the high water mark. Consider what happens when an adversary sends a forged packet with source address s, key counter `0xFF`, frame counter `0xFFFFFFFF`, and any payload whatsoever (not necessarily a valid ciphertext under key k). The recipient will decrypt the packet under key k, resulting in random garbage. Since there is no access control or message authentication, the recipient will accept the packet even though it contains garbage. However, before passing the garbled payload to the application, the media access control layer will update the high water mark to `0xFFFFFFFFFF`. The next time the real sender s tries to send a legitimate packet, the recipient will reject it no matter what s does, because the high water mark has reached its maximum value and any subsequent packet will appear to be replayed. Similar attacks can also be used to prevent delivery of the next n packets to be sent on some link, where the number n can be selected by the attacker.

This shows that an attacker can permanently disrupt a 802.15.4 link, if that link uses AES-CTR with replay protection enabled. The attack is easy to mount, because it only requires sending a single forged packet; the attacker needs no special access or equipment.

3.3.3 No Integrity on Acknowledgment Packets

The 802.15.4 specification does not include any integrity or confidentiality protection for acknowledgment packets. When sending a packet, the sender has the option of requesting an acknowledgment from the recipient by setting a bit in the flags field. If the acknowledgment request flag is set, the recipient returns an acknowledgment packet that contains the packet's sequence number. The sender's media access control layer resends the packet a finite number of times if it doesn't receive the acknowledgment in time. The sending application is notified when the acknowledgment arrives.

However, the lack of a MAC covering acknowledgments allows an adversary to forge an acknowledgment for any packet. An adversary need only create the forged acknowledgment with the appropriate sequence number from the original packet; this is not hard, since this sequence number is sent in the clear.

This weakness can be combined with targeted jamming to prevent delivery of selected packets. Suppose an attacker identifies a packet that he wishes to ensure is not received by the intended recipient. The attacker can transmit a short burst of interference while the packet is being sent, causing the CRC to be invalid at the recipient and the packet to be dropped by the recipient. Then, the attacker can forge a valid-looking acknowledgment, fooling the sender into thinking that the packet has been received.

This vulnerability renders acknowledgments untrustworthy when adversaries are present. For example, suppose that an application uses the acknowledgment as a part of a reliable send interface. The application keeps trying to send the packet as long as it remains unacknowledged. If the sending application receives an acknowledgment, it can never be sure whether the data *actually* arrived at the destination. The acknowledgment might be legitimate, or it might be forged. Consequently, acknowledgments can be used as a hint for performance improvement, but they should not be relied upon.

4. RECOMMENDATIONS

In Section 3, we outlined a number of deficiencies in the 802.15.4 specification. These problems have a number of solutions. Some can easily be avoided by an application programmer if they are aware of the problem. Others may require the radio designer to adjust their designs. Alternatively, the 802.15.4 standards committee could solve these problems in the next revision of the 802.15.4 specification. In fact, fixing some of the problems may require cooperation from more than one party. In this section, we suggest workarounds that can be used to defend against the weaknesses described earlier.

4.1 Application Designers

Don't Use AES-CTR Security Suite. As we noted, the AES-CTR can lead to problems in protocols. The inability to properly support replay detection (despite the specification's claims to the contrary) is just one example. We do not recommend that any application use the AES-CTR security suite.

Don't Rely on Acknowledgments. Application designers should not rely on the acknowledgments as they stand since they don't provide any guarantees. Receiving an acknowledgment doesn't mean the recipient actually received the packet.

If an application needs the functionality of acknowledgments, they should use application level acknowledgments by sending data packets with integrity control and not use the acknowledgments that 802.15.4 provides. Application level acknowledgments duplicate much functionality that the media access control layer provides. For example, an application would need to use its own mechanism to retry a send until an application level acknowledgment arrives. There is certainly added complexity in this workaround, but barring integrity protection for acknowledgments we believe it is the only safe alternative.

4.2 Hardware Designers

Include Support for 255 ACL Entries. To properly support pairwise keying, hardware designers should include support for many ACL entries. Proper support for individual keying in large networks demands many ACL entries.

Retain ACL List in Low Power Mode. As noted in Section 3.1.2, some hardware implementations may reset the contents of the ACL list when entering low power mode. We advocate that the radio should retain the contents of the ACL and its associated security material even when in low power mode.

Reserving a block of nonce values can help when power is removed completely. The radio chip can reserve, for example, 256 nonce values at a time by writing the largest value to the flash. After recovering from power loss, the radio chip uses the next larger block of nonce values than what is stored in the flash.

Expose Nonce for each Received Packet. The replay detection mechanism is not workable when a cryptographic key is between more than two nodes. The default ACL entry poses problems for replay detection, because it is used for incoming packets from many sources but only has state for a single counter in the ACL entry. This means that hardware replay detection, as specified, cannot be meaningfully be applied to the default ACL entry. The radio chip should offer support for applications to implement their own replay detection when they need the functionality of the default ACL entry with replay detection.

To support application level replay detection when using group keys, the radio should expose the replay counter value when signaling a packet's arrival to higher layers. Currently, the incoming frame and key counter (recall their concatenation is the replay counter) are stripped from incoming messages so that higher layer applications are not aware of them. By exposing the replay counter value, applications could manage their own counter high water marks per each sender. This allows the recipient to implement replay protection when a key is shared among a group of nodes.

Eliminate Support for the AES-CTR Security Suite. Since AES-CTR is an unauthenticated suite, and since unauthenticated suites have major weaknesses, we urge radio designers to leave this suite unimplemented. Fortunately the

802.15.4 standard specifies that AES-CTR support is optional, so leaving it unimplemented will not endanger standards compliance. Additionally, AES-CTR does not provide sequential freshness, as the specification claims.

Removing support for the AES-CTR will prevent receiving any messages sent with the AES-CTR security suite, which is perhaps undesirable when different vendors' radios need to communicate. As a compromise, we suggest that radio designers remove the capability for sending in the AES-CTR suite and advise users not to use the suite.

Leaving this unimplemented will help prevent misuse and attacks in the future.

4.3 Specification Writers

Warn against Dangerous ACL Configurations. The specification should require radios to warn applications when they attempt to set up a dangerous ACL configuration. This includes using two ACL entries with the same address, or more worryingly two different addresses with the same key. It is a policy decision whether to even allow the latter case since it leads to confidentiality violations so easily.

Better Support for Keying Models. As noted in Sections 3.1.1 and 3.2.1, 802.15.4 is most suited to the pairwise key model. We advocate changes in the specification to more easily support different keying models. For example, decoupling nonce storage (used for sending packets) from replay counters (used to receive packets) can better support the other keying models we mentioned.

Remove support for the AES-CTR Suite. We recommend that the specification writers eliminate the possibility of using the AES-CTR security suite. It is the only unauthenticated mode of operation, leading to application protocol level vulnerabilities. Additionally, replay protection is not secure with this suite. We do not think that it should be a part of the specification.

Support Authenticated Acknowledgments. We recommend that specification writers add the option of authenticated acknowledgments. These acknowledgments would only be possible to generate by a node that shares a cryptographic key with the sender.

For example, instead of including the 1 byte sequence number from the original packet, the recipient can use a 4 or 8 byte MAC over a buffer comprised of the received packet and its own address. The sender computes this MAC value before sending the packet and saves it. The sender then compares the stored value with the value in the acknowledgment packet. It only accepts the acknowledgment as valid if it receives the expected MAC value; an adversary cannot compute the MAC value since it does not have access to the data packet and the private cryptographic key that the sender shared with the recipient.

Eliminate difference between Key and Frame Counter. Finally, as a nomenclature suggestion, we would advocate removing the distinction between the key and frame counter and let the hardware manage the entire counter. If the user wants to continue using the key after losing the nonce state, they will need to store at least the key counter. However, to effectively use the nonce space, the application should also store the frame counter as well. Not saving the nonce state, then, only permits 255 losses of power.

By eliminating the distinction between the key and frame counter, no functionality is lost, while simplifying the specification and use of the security package: the application does not need to include special logic to periodically increment the key counter.

5. CONCLUSION

We have outlined a number of problems and pitfalls when using the 802.15.4 specification. These problems can lead to undetected security vulnerabilities in deployed applications.

Despite the presence of these defects, the 802.15.4 security architecture is sound. It includes many well designed security features and presents a step forward for embedded device wireless security. Proper use of the security API, with an awareness of its subtleties, can lead to secure applications. We have suggested ways that application developers, radio designers, and specification writers can work around or remedy the problems we have described.

6. ACKNOWLEDGMENTS

Rob Johnson, Chris Karlof, and Joe Polastre provided invaluable feedback on earlier drafts of this work.

7. REFERENCES

[1] Zigbee alliance. `http://www.zigbee.org`.

[2] Weak crc allows packet injection into ssh sessions encrypted with block ciphers. Computer Emergency Response Team (CERT), June 1998. VU 13877.

[3] Smart buildings admit their faults. Lab Notes: Research from the College of Engineering, UC Berkeley, `http://www.coe.berkeley.edu/labnotes/1101smartbuildings.html`, November 2001.

[4] Weak crc allows last block of idea-encrypted ssh packet to be changed without notice. Computer Emergency Response Team (CERT), January 2001. VU 315308.

[5] Weak crc allows rc4 encrypted ssh1 packets to be modified without notice. Computer Emergency Response Team (CERT), January 2001. VU 25309.

[6] Wireless medium access control and physical layer specifications for low-rate wireless personal area networks. IEEE Standard, 802.15.4-2003, May 2003. ISBN 0-7381-3677-5.

[7] Atmel at86rf210 z-link transceiver data sheet. `http://www.atmel.com/dyn/resources/prod_documents/doc5033.pdf`, 2004.

[8] Chipcon cc2420 data sheet. `http://www.chipcon.com/files/CC2420_Data_Sheet_1_1.pdf`, 2004.

[9] Motorola mc13192 datasheet. `http://e-www.motorola.com/files/rf_if/doc/data_sheet/MC13192DS.pdf`, 2004.

[10] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation. In *Proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS 97)*, 1997.

[11] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The Security of the Cipher Block Chaining Message

Authentication Code. *Journal of Computer and System Sciences*, 61(3):362–399, December 2000.

[12] Steven M. Bellovin. Problem areas for the IP security protocols. In *Proceedings of the Sixth Usenix UNIX Security Symposium*, 1996.

[13] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting mobile communications: The insecurity of 802.11. In *MOBICOM*, 2001.

[14] G.L. Duckworth, D.C. Gilbert, and J.E. Barger. Acoustic counter-sniper system. In *SPIE International Symposium on Enabling Technologies for Law Enforcement and Security*.

[15] V. Rijmen J. Daemen. The Block Cipher Rijndael. In J.-J. Quisquater and B. Schneier, editors, *Smart Card Research and Applications, LNCS 1820*, pages 288–296. Springer-Verlag, 2000.

[16] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, and David Culler. Wireless sensor networks for habitat monitoring. In *First ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.

[17] Robert Szewczyk, Joseph Polastre, Alan Mainwaring, and David Culler. Lessons from a sensor network expedition. In *First European Workshop on Wireless Sensor Networks (EWSN '04)*, January 2004.

[18] Matt Welsh, Dan Myung, Mark Gaynor, and Steve Moulton. Resuscitation monitoring with a wireless sensor network. Supplement to Circulation: Journal of the American Heart Association, October 2003.

[19] D. Whiting, R. Housley, and N. Ferguson. Counter with cbc-mac (ccm). RFC 3610, September 2003.