MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

# A Security Design for a General Purpose, Self-Organizing, Multihop Ad Hoc Wireless Network

Messerges, T.; Curkier, J.I.; Kevenaar, T.A.M.; Puhl, L.; Struik, R.; Callaway, E.

TR2003-114 October 2004

**Abstract**

We present a security design for a general purpose, self-organizing, multihop ad hoc wireless network, based on the IEEE 802.15.4 Low-Rate Wireless Personal Area Network standard. The design employs elliptic-curve cryptography and the AES block cipher to supply message integrity and encryption services, key-establishment protocols, and a large set of extended security services, while at the same time meeting the low implementation cost, low power, and high flexibility requirements of ad hoc wireless networks.

*SASN 2003*

# A Security Design for a General Purpose, Self-Organizing, Multihop Ad Hoc Wireless Network

Tom Messerges*, Johnas Cukier**, Tom A.M. Kevenaar***, Larry Puhl*, Rene Struik****, and Ed Callaway*

* Motorola Laboratories; {tom.messerges, larry.puhl, ed.callaway}@motorola.com
** Mitsubishi Electric Research Laboratories; jic@merl.com
*** Philips Electronics; tom.kevenaar@philips.com
**** Certicom Corporation; rstruik@certicom.com

**Abstract**
We present a security design for a general purpose, self-organizing, multihop ad hoc wireless network, based on the IEEE 802.15.4 Low-Rate Wireless Personal Area Network standard. The design employs elliptic-curve cryptography and the AES block cipher to supply message integrity and encryption services, key-establishment protocols, and a large set of extended security services, while at the same time meeting the low implementation cost, low power, and high flexibility requirements of ad hoc wireless networks.

## 1 Introduction

Until recently, research on self-organizing wireless ad hoc networks mainly focused on routing approaches and the problems of data transfer over a network with dynamically changing topology. Stajano and Anderson [1] were among the first to realize the importance of security in such networks and described the most important security issues, illustrated by some real-life examples. Later publications arrived at the same conclusion [2], [3]. The very nature of ad-hoc networks and the cost constraints that are often imposed on these make these networks difficult to secure. Communications cannot rely on the online availability of a fixed infrastructure, thus necessitating decentralized online key management, rather than centralized key management, such as implemented with the well-known 802.11 WLAN standard. Furthermore, ad hoc networks may be highly versatile, involving short-lived communications between devices that may never have met before, thus complicating initial trust establishment. Well-designed security services can contribute to the reliability and robustness of the network's communication infrastructure and to the protection of application data sent over this network. Recent security research for ad hoc networks seemed to focus on distributing the role of the Certifying Authority over some or all devices in the network [4]-[9], the main approach being based on threshold cryptography [10] and allowing specific coalitions of devices to act together as a source of trust such as a certificate authority (e.g., to generate public-key certificates). Unfortunately, most of these approaches do not seem to be very efficient, either in terms of computational or communication overhead. Further, even if efficient methods to share trust were available, this only partially solves security issues that may arise in constrained ad hoc networks.

In this paper, we describe a security architecture and some implementation aspects thereof, targeted at a low data rate, multihop wireless ad-hoc network, where we focus on architectural choices that allow an efficient implementation and a low communication overhead.

# 2 The network and its security needs

## *2.1 Description of the Wireless Network*

There are many applications, such as industrial control and monitoring, asset tracking, intelligent agriculture, home automation, and consumer electronics, for which conventional communication networks are unsuitable, due to excessive power consumption, high implementation cost, or the need to employ some type of trained network administrator for network establishment or maintenance. Self-organizing ad hoc wireless networks can overcome these limitations, but introduce security issues of their own that must be satisfactorily addressed to ensure successful deployment of these networks in the market.

Self-organizing ad hoc wireless networks suitable for the applications listed above have several performance metrics that differ from those of a conventional communication network, such as an IEEE 802.11b WLAN. The typical performance metric used for a WLAN is data throughput: the higher the throughput, the better the WLAN. For the ad hoc wireless networks we consider, however, major metrics are low implementation cost (a few dollars), very low average power consumption (roughly 10 to 100 μW), minimum need for network administration, and the flexibility to service a very wide range of applications. The emphasis on cost and power savings at the expense of network data throughput places important boundary conditions on the security architecture, since computational resources (RAM, ROM, and processor speed) are severely limited. The requirement that the network be suitable for a wide range of applications places great demands on the flexibility of the security system, since it must be capable of supplying security services that may be tailored towards a wide variety of network topologies and application requirements, rather than being fixed.

The protocol stack of a network device is shown in Figure 1. The network employs the IEEE 802.15.4 Low-Rate Wireless Personal Area Network (LR-WPAN) [11] standard. This standard specifies the physical (PHY) layer and medium access control (MAC) sublayer of a low cost, low power consumption, ad hoc wireless network. The higher layers built on top of the IEEE 802.15.4 layers include a network layer, an application support layer, application endpoints, and security services.

The PHY layer actually presents a choice between two different layers; the implementer may choose operation in the 2.4 GHz ISM band, or operation in the 868 MHz band (in Europe) and the 915 MHz band (in the Americas). The standard specifies a raw data rate of 250 kb/s in the 2.4 GHz band, 20 kb/s in the 868 MHz band, and 40 kb/s in the 915 MHz band. Since it is a WPAN, the range of an individual node may be limited; however, the physical area covered by the network may be greatly extended by the use of multihop routing.

The MAC layer provides services that higher layers can use to access the physical radio. MAC layer protocols provide a means for reliable, single-hop communication links between devices. These protocols include special beaconing and sleep mode functions that enable low duty cycles and ultra-low power consumption – a vital concern for battery-operated network devices. To meet its low implementation cost and low power consumption goals, the IEEE 802.15.4 standard specifies a worst-case maximum MAC payload length of 102 bytes. Since message fragmentation is expensive, both in terms of implementation hardware (buffers, etc.) and power consumption, this limit makes it advantageous to keep exchanged messages below this length. IEEE 802.15.4 supports unicast (i.e., one-to-one) and broadcast (one-to-all) messages, but does not natively support multicast (one-to-many) messages.

The network layer provides routing and multi-hop functions needed for creating different network topologies, such as peer-to-peer, star, cluster tree, and mesh structures. Multi-hop communication capabilities increase the effective range of a device, allowing it to reduce transmission power and save battery life.

On top of the network layer sits the application support layer, which provides a solid foundation for constructing applications. All applications, from lighting to home security to toys and games, will use the same basic structure for over-the-air commands. Multiple applications (i.e., application endpoints) can reside on a single network device. These applications share a single radio, and the application support layer helps ensure that they will interoperate with each other.

Alongside all of the layers sits the so-called security toolbox, which is the main focus of this paper. The IEEE 802.15.4 MAC layer provides means for securing the data communications themselves, but does not provide facilities for key establishment and maintenance. These services, the end-to-end protection of data communicated over multiple hops, and the enforcement of security policies, are left to the security toolbox.

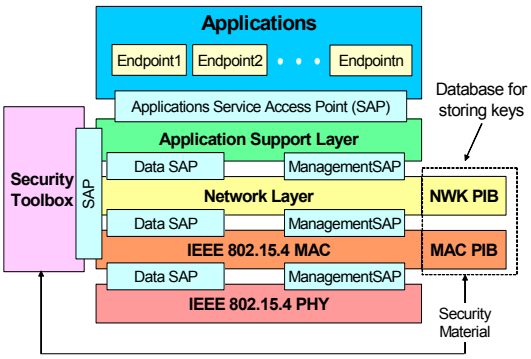


**Figure 1 The Protocol Stack**

## *2.2 Security Needs*

The ad hoc wireless network described in this work is designed to offer standard solutions for a diverse range of products. Products may have different security requirements or none at all. Ultimately, cost, performance, complexity, flexibility, and ease-of-use are all factors that determine the security requirements of a particular application. Hence, the need for flexibility in designed security services. The "security toolbox" approach satisfies this need for flexibility: It includes all the security services foreseen to be useful to potential products. Application designers and implementers will pick and choose services from this toolbox, but will not have access to the protocols underlying these services.

When deciding which tools and services to include into this toolbox, a few representative applications were considered, including residential lighting, industrial control and building automation, and autonomous sensor networks. These application areas are described below.

### Residential Lighting

In a typical residential application a consumer might visit a home improvement store to buy a wireless dimmer switch and a wireless light socket (e.g., Figure 2). The dimmer switch would be battery powered and the light socket would be mains powered. The fact that each device has an identifying label, even though different companies may manufacture them, gives the consumer assurance that the two devices will interoperate. The consumer will install the devices by mounting the dimmer switch on the wall and screwing the light socket onto his lamp. Next, the

switch and socket would be paired together by manually pressing a button on each device. This would cause the devices to activate, detect each other, and establish a communication link. In the future, the customer might want to add more devices to this network. For example, he might want two lamps controlled by the same switch, he might want an extra switch added to another location, or he might want to add another type of device, such as a ceiling fan, home alarm system, sensors, or a thermostat.

Most residential applications require minimal security, but we still do not want a neighbor's device (or a hacker) to gain control of our system. Hence, one needs logical separation of networks. In the simplest case, the dimmer switch and lamp socket will establish a shared secret key, a so-called "link key", which will be used to ensure the socket that it executes only commands originating from the dimmer switch and not from, e.g., one of the neighbor's devices or a hacker's device. In more complex cases, keys shared by two devices may also need to be shared with other devices (e.g., to support multi-cast), a new key may need to be automatically reestablished (e.g., for easy reconfiguration when the battery in the dimmer switch is replaced), or the key may need to be refreshed (e.g., to support changes to the network, such as the addition or removal of a device). The security toolbox needs to support all of these security functions.

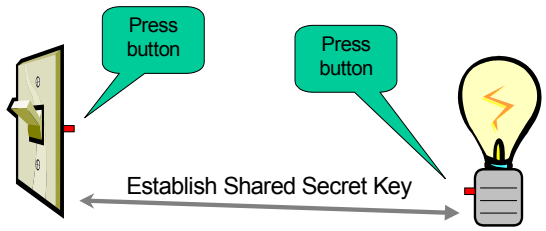


**Figure 2 Manual Configuration (Residential Setting)**

### Industrial Control and Building Automation

In an industrial setting, a contractor may decide to use wireless devices to save on the cost of wiring a large building. Many hundreds, and even thousands, of lights and controls for offices, conference rooms, hallways, warehouses, etc. need to be configured. Pressing buttons to manually configure such as large system is impractical, so an alternate method is needed. In this case, a configuration device, such as a portable computer, is used to virtually wire the building by linking switches to lights, binding sensor outputs to alarms, connecting thermostats to the ventilation system.

This configuration device may also function as a security manager (e.g., Figure 3). A security manager is capable of installing initial security material into devices to help create trust relationships and allow for the future establishment of link keys. Building maintenance personnel might use configuration devices to override, repair, or control already-configured devices. In industrial settings, such as these, the configuration device needs a secure method to identify itself to the other devices and the other devices need to be capable of receiving security information from it. The security toolbox will have services to support security manager functions. Applications will only implement capabilities that are needed.

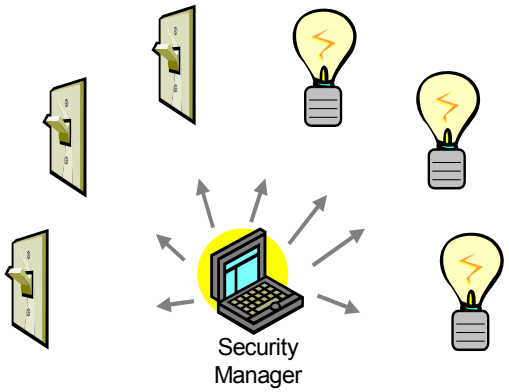


**Figure 3 Assisted Configuration (Industrial Setting)**

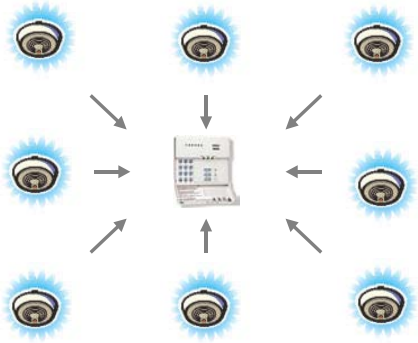

**Figure 4 Autonomous Configuration (Sensor Networks)**

### Autonomous Sensor Networks

In some situations, the setup of wireless network devices needs to be either partially or entirely autonomous. For example, a large warehouse owner may wish to install smoke detectors throughout his building (e.g., Figure 4). An economical way to do this is to buy one central controller device and many individual smoke detectors. Instead of using a security manager to configure this system, the warehouse owner simply mounts the detectors throughout his building and allows the devices to self-configure. The devices would automatically discover each other, detect communication routes to the central controller, and create a secure mesh network (e.g., establish link keys). The installation is kept simple because a separate configuration device that serves as security manager is not needed. However, securing this autonomous network presents unique challenges. For example, the devices in the network must be capable of identifying each other and determining whether other devices should be allowed into the network. Routing protocols in the network need to be secured and end-to-end security between each detector and the central controller needs to be ensured. One way to accomplish a secure mesh network is to use a shared group key that is pre-loaded into each device. However, a common group key poses a security risk if any one device is compromised. The use of asymmetric keys (i.e., public/private keys) along with digital certificates to establish individual link keys can help reduce this risk. Public-key based techniques, along with digital certificates, may help to mitigate this risk, since these restrict the impact of key compromise to the compromised node itself, rather than to all its key-sharing parties. The actual deployment of public-key based techniques depends on a cost/benefit trade-off.

# 3 Security Design

## *3.1 Assumptions*

In order to design security for these wireless network devices, certain assumptions are made. One of the first assumptions we make is that a device will not intentionally or inadvertently transmit its keying material to other devices unless the keying material is protected using secure mechanisms (e.g., during a key-transport operation). However, physical access to a device may allow access to keying material. That is, keys are kept secret, but this secrecy is not necessarily guaranteed with tamper-resistant hardware.

Another basic assumption is that the security software and hardware operate as expected. For example, the key-establishment software properly executes a protocol to derive a new key and a hardware random number generator is sufficiently random.

In addition, due to the cost constraint mentioned above, we must assume that different applications using the same radio are not logically separated using a firewall or sandbox. Moreover, from the perspective of a given device it is not possible to verify whether cryptographic separation between different applications on another device, or even between different layers of the communication stack hereof, is indeed properly implemented. Hence, we have to assume that application endpoints - separate applications using the same radio - trust each other (i.e., there is no cryptographic task separation). Also, lower layers (e.g., NWK or MAC) are fully accessible by any of the application endpoints. These assumptions lead to an open trust model for a device: different layers of the communication stack and all applications running on a single device trust each other. However, by design of the security toolbox Service Access Point (SAP), no layer has direct access to the security protocols running in the security toolbox. Outside layers may merely request security services and receive results from them.

## *3.2 Security Architecture*

Based on the assumptions in the previous section we will make the following architectural choices. First, we establish the principle that “the layer that initiates a message is responsible for securing it.” For example, NWK-layer commands are secured by calls to the security toolbox from the NWK layer, and do not undergo further security processing at the MAC layer prior to transmission. This contributes to an efficient implementation by eliminating duplicate security operations (such as message integrity code calculations) at each layer of the protocol stack and it reduces the security communication overhead to one integrity code per message.

Second, security is based on the premise that a single, unique, link key is established for each pair of devices that wish to securely communicate. This limits the amount of keying material that has to be stored on a device and hence implementation costs.

The architecture includes security needs at three layers of the protocol stack. The MAC and NWK layers need services for securing MAC frames and NWK frames, respectively. The application support layer needs services for establishing and managing security relationships (e.g., link keys and group keys). Here, MAC frames are delivered from one device to another via a single hop and NWK frames are delivered from one device to another via one or more hops. The security toolbox provides services to the MAC and NWK layers to protect these frames. Although applications are responsible for selecting their own security protection level, they will delegate the actual cryptographic processing to the application support and NWK layers. This design makes sure that applications do not need direct access to the security toolbox itself, thus realizing considerable separation of concerns.

### MAC Layer Security Services

Security at the MAC layer can prevent MAC data and MAC commands from being compromised, stolen, replayed or otherwise modified. The IEEE 802.15.4 standard offers three choices: no security, Access-Control List (ACL) security, or cryptographic security. When using ACL security, a device maintains a list of devices from which it expects to receive communications. When the MAC layer receives a frame, it notifies the upper layer that the frame was received and also indicates whether the sender of the frame is in its ACL.

When using cryptographic security, the MAC layer uses the Advanced Encryption Standard (AES) [12]. The MAC layer can be configured to send messages that are encrypted, appended with integrity bits, or both. The number of integrity bits can be set to 32, 64, or 128. When the MAC layer receives a frame that is cryptographically protected, it looks at the source of the frame and retrieves the link key associated with that source from a database. It then uses this link key to check any integrity bits and decrypt (if encrypted) the payload. The MAC layer then notifies the upper layer that a message was received and it reports the results of these security operations.

Encryption at the MAC layer is done using AES in counter mode and integrity is done using AES in Cipher Block Chaining (CBC) mode [13]. Both encryption and integrity is done using a combination of counter and CBC modes (i.e., CCM [14]). Figure 5 shows a MAC frame with security. Security adds a frame count, a key sequence count, and integrity field to the MAC frame. The entire frame is integrity protected, but only the payload is encrypted. The frame and key sequence counts are included to prevent replay attacks. Sending devices will increase these counts with every message sent and receiving devices will keep track of the last received count from this sending device. If a message with an old count is detected, it is flagged with a security error.

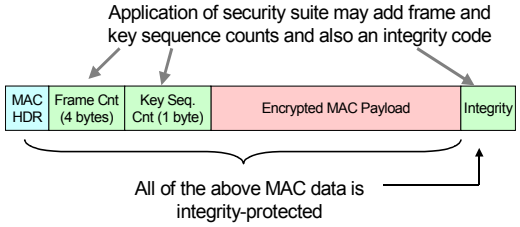


**Figure 5 MAC Command Frame with Security on the MAC level**

### Security Toolbox Services

The security toolbox services augment the MAC layer security in a number of ways. The MAC layer specification leaves the establishment and maintenance of link keys to the upper layers and only handles security for a single hop. For example, the MAC layer does not know how to properly handle information for routing that is part of a NWK frame header. If only MAC layer security were used, NWK headers would be encrypted because they are part of the MAC payload. Each intermediate node would need to decrypt a frame to obtain the information needed for routing. End-to-end security would not be achieved. Moreover, each intermediate node would have to implement security, even if it were installed only to provide a message relaying function for other network nodes.

As shown in Figure 6, the NWK frame is protected with a scheme similar to that used for MAC frames. The NWK header is not encrypted, so routing information can be used across multiple hops without the need to reapply security or the need to trust every device at intermediate hops. Note that this use of end-to-end security, where one only protects information that does not

change on a hop-by-hop basis, allows the implementation of routing to be independent of security considerations.

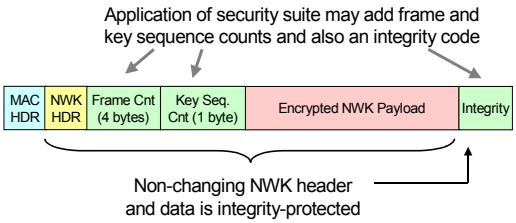


**Figure 6 NWK Command Frame with Security on the MAC level**

### Application Layer Security

A single radio may support multiple applications. Some maintain that, in this situation, the application layer would need to provide individual security for each of these applications. However, one of our basic assumptions is that separate applications using the same radio trust each other (e.g., there is no task separation). This assumption leads to the result that the application layer can rely on the NWK layer for securely sending its messages. Therefore, although the application layer may initialize or configure security parameters, the protection of frames is only performed at the NWK or MAC layers. This design decision results in a more efficient implementation. Secured APL frames have a format very similar to the one depicted in Figure 6.

A meter reader device is an interesting example that shows the need for multiple secure applications on a single network device. In this example, a metering device would have concurrent applications running for the water, gas, and electric companies. These applications would be allowed to communicate only with their respective owners (i.e., only the electric company can read the electric meter, only the gas company can read the gas meter, etc.). To enforce this security policy each application would set up its own link keys. A metering device (attached to the home) would have keys for each of the applications (e.g., water, gas, electric), but meter reading devices (e.g., the water company's truck, the gas company's truck, etc.) would only contain their individual keys.

### Device Roles

Figure 7 shows that device types are modeled using three tiers: a physical type, a logical type, and an application type. At the lowest tier, the IEEE 802.15.4 standard defines two physical device types, a Full-Functional Device (FFD) and a Reduced-Functional Device (RFD). An RFD can take on the logical role of an end device, while an FFD can also take on the role of coordinator (see Figure 7) or router. RFDs will have less computational and memory capacity than FFDs. Implementing security solutions in an RFD poses the biggest challenge, due to strict implementation constraints. An RFD is likely to be a very low-end processor (e.g., an 8051 or HC08) clocked at 4 to 12 MHz. Less than 5 Kbytes of program memory (ROM) and 256 bytes of RAM are available for security and there is potentially no non-volatile (FLASH) memory for storing keys.

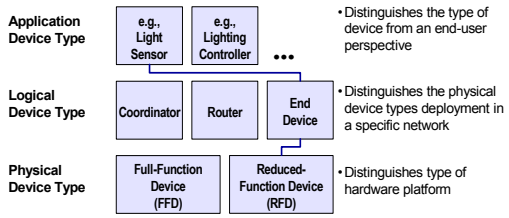


**Figure 7 Device Model**

Above the physical device types, we define three logical device types: a coordinator, a router, and an end device. There is exactly one coordinator per network; the coordinator identifies the network and defines many of the network parameters. Coordinators and routers can connect to routers and end devices, but end devices cannot connect to other end devices. An example network topology showing these logical device types is given in Figure 8.

From the end-user perspective, the device type represents a particular application (e.g., switches, lights, sensors, thermostats, etc.).

There are also important security roles to consider, for example, a security manager, a Certificate Authority (CA), and a Key Distribution Center (KDC).

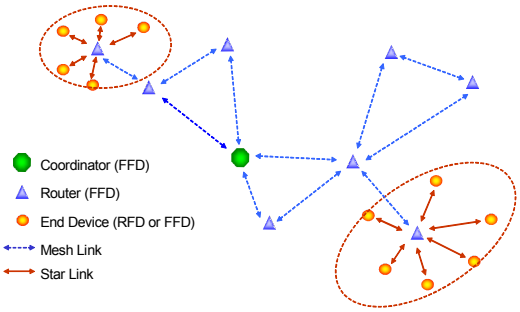


**Figure 8 Example Network Topology**

*Security manager*

A device with security manager capabilities is used to configure a network by provisioning initial trust data to network devices. The initial trust data tells a device which devices to trust and can include device addresses, master keys, public keys, or certificates. Devices in the network use initial trust data to establish a link key, which is then used for secure communications. A further decomposition of functionality is possible, by separating the tasks of configuration manager and security manager (details are beyond the scope of this paper).

End devices need the capability to recognize (i.e., authenticate) a security manager before accepting initial trust data. Authentication of a security manager can be done using cryptographic means (e.g., public key) or physical means (e.g., a cable). Also, a security manager is a portable device and, as such, may not always be kept physically secure. A system needs to be designed so that it can recover when a security manager is stolen or lost. A security manager may also be used as a proxy for relaying initial trust data from a CA or KDC.

*Certificate authority and key distribution center*

A Certificate Authority or Key Distribution Center is also capable of provisioning initial trust data. The CA or KDC is kept physically secure and is not portable. So, compared with a security manager, it should be less likely that a CA or KDC will be stolen, lost, or under physical control of an attacker. In keeping with traditional definitions, a CA distributes initial trust data for public-

key based systems (e.g., private keys, public keys, root keys, and certificates) and a KDC distributes initial trust data for symmetric-key based systems (e.g., a master key).

In order to establish authenticated communications, network devices using our architecture need to be pre-configured with initial trust data. Initial trust data allows a device to recognize other devices it should trust. A trust relationship can be based on different types of information. For example, a rudimentary level of trust can be achieved using a filtering mechanism based on the unique 64-bit addresses required in every device implementing the IEEE 802.15.4 standard. Two devices might have a non-cryptographic way of establishing the identity of the other device. An example hereof is "pushing buttons", where a human operator controls which devices are executing a protocol. Anyway, from a security viewpoint a secure device will have to go through some enrollment phase during which it receives its initial trust data.

Our view is that the primary security relationship exists between pairs of devices. This means there will be a single invocation of a key-establishment mechanism per pair of devices that want to establish a link key. The initial trust data is used when deriving this link key. The security toolbox provides four key-establishment protocols for establishing link keys.

### *3.3 Key-Establishment Protocols*

In our design, two network key-establishment protocols are based on symmetric-key cryptography and two are based on public-key cryptography. While public-key techniques offer greater resistance to some forms of attack, symmetric-key techniques are employed for applications that cannot afford the cost and complexity of public-key implementations. Figure 9 shows features shared by all key-establishment protocols. Each protocol involves two entities, an initiator device and a responder device, and each involves four steps, the establishment of a trust relationship, the exchange of ephemeral data, the use of this ephemeral data to derive a link key, and the confirmation of the link key.

*Symmetric-key key establishment*

The two symmetric-key protocols we propose for establishing link keys are the Symmetric-Key Key Establishment (SKKE) protocol and the Unprotected Key Establishment (UKE) protocol. In the SKKE protocol, a trust relationship is established using a shared, secret, and symmetric key, referred to as a master key. This master key, for example, may be pre-installed during manufacturing, may be derived using some cryptographic method, may be installed by a KDC, may be based on user-entered data (e.g., PIN, password, or key), or may be read off the package or chassis of the wireless product (e.g., a bar code). The secrecy and authenticity of the master key needs to be upheld in order to maintain the trust foundation of our SKKE protocol.

In the UKE protocol, the master key is a fixed, default value. In this case, potential adversaries may know the master key. The trust relationship for the UKE protocol is based merely on knowledge of the other device's IEEE 64-bit address, which could be obtained from initially exchanged messages. This approach does not give any cryptographic security on the derived link key because it is susceptible to eavesdropping. Nevertheless, the UKE protocol might be useful in settings where this risk is limited during execution of the key establishment protocol. It is especially suitable for low-cost implementations that cannot change a master key (e.g., systems without flash or other nonvolatile programmable memory).

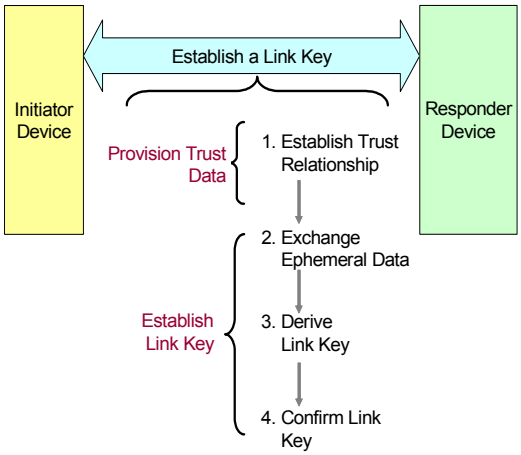


**Figure 9 Four Steps for Key Establishment**

Successful completion of either the UKE or SKKE protocols results in the following:

- Both devices share a link key.
- Key confirmation: Each device knows that the other device has computed the correct link key.
- No unilateral key control: No device has complete control over the link key that is established.
- No forward secrecy: Eavesdropping during the UKE protocol or eavesdropping and compromise of the master key in the SKKE protocol exposes all the future and past communications.

If the SKKE protocol is successfully used and each device knows beforehand that only the other device possess the master key, then implicit key authentication is achieved (i.e., each device is sure that no other device has access to the established link key). If the SKKE protocol is used without each device knowing who has access to the master key (also true for the UKE protocol), then implicit key authentication can still be achieved provided that no eavesdropping occurs (i.e., no passive attacks), that no messages are modified (i.e., active attacks), and that the protocol is executed in an environment where the two devices have a non-cryptographic way of establishing the identity of the other device.

The building blocks required for the UKE and SKKE protocols include the AES block cipher, an unkeyed hash function (e.g., the Matyas-Meyer-Oseas hash [15] based on AES), and a random number generator. For maximum hardware and software reuse, the random number generator itself may employ the AES algorithm.

*Public-key key establishment*

The two public-key protocols we propose are the Public-Key Key Establishment (PKKE) protocol and the Certificate-Based Key-Establishment (CBKE). The PKKE protocol (which does not employ certificates) can be used to prevent passive attacks that involve an eavesdropper monitoring the key-establishment procedure. Trust between devices is established by exchanging unsigned public keys together with the corresponding device IDs. With this method, one cannot check cryptographically to whom a public key belongs and hence one should rely on the environment to provide assurance of the trustworthiness of a public key and purported device ID. The PKKE protocol is especially useful for network applications which lack the ability to employ a CA.

The Certificate-Based Key-Establishment (CBKE) protocol uses public-key technology with (digital) certificates and root keys. A digital certificate is simply a public key together with the

device's 64-bit IEEE address, signed by the CA. Certificates provide a mechanism for checking cryptographically to whom the public key belongs and if the device is a legitimate member of a particular network. Once a certificate and root key are securely provisioned to a device, active and passive attacks in subsequent key establishment protocols can be thwarted. We propose the use of 163-bit Elliptic-Curve Cryptographic (ECC) techniques for the PKKE and CBKE protocols. ECC techniques offer a reasonable computational load (especially since they will be performed infrequently in most application scenarios), and smaller key lengths for equivalent security than other techniques. Small key lengths are important to minimize the size of message storage buffers on network devices, which in turn reduces their implementation cost.

Both the symmetric-key based and the public-key based protocols were designed such as to maximize commonalities between message flows and to allow re-use of cryptographic building blocks. This way, a single implementation of telecommunication overhead and error handling seems to be possible. Furthermore, each protocol step, including those for elliptic-curve based key establishment with certificates, could be implemented within a single frame, due to their small length of less than 100 bytes.

### *3.4 Other Basic Security Services*

In addition to key establishment, our security suite also provides basic services for key transport, entity authentication, and frame protection. Key transport services are needed when keys are delivered via a security manager, sent to or restored from a backup device, distributed to a group of devices, or refreshed. Entity authentication services are needed to detect whether a device is still available (e.g., memory can be freed by deleting keys of devices that are no longer available). Frame protection services are needed to provide the NWK layer with capabilities to protect NWK frames that are delivered over multiple hops.

### *3.5 Extended Security Services*

The basic security services can be used to construct extended services that involve security policies. Services that need to be secured include key updates, orphan management, assisted association, remote association, portability, coordinator linking, indirect addressing, multicast, code download, and service discovery.

*Key update*

Keys may be updated as part of a security policy or to recovery from a lost key. In our implementation, a mandated security policy is that keys shall be updated when the key or frame counts (in the MAC or NWK headers) reach their maximum value (since otherwise freshness or even confidentiality would be at jeopardy) or when a device leaves or enters a group that is sharing a key. Expiration times may also be used to cause a key update.

*Orphan management*

End devices (i.e., RFDs) need to be kept extremely simple and cheap. To meet this requirement, end devices may not have expensive features such as nonvolatile memory (e.g., flash memory). When power is removed, as may occur during a power outage or due to a discharged battery, link keys and other configuration information stored in RAM will be lost. A device that loses its memory, referred to as an "orphaned device", will need to automatically reestablish its association with a router in a secure fashion. User intervention should not be required for retrieving orphaned devices back into the network. An example of such a scenario is that with remote controls, where the simple act of replacing a battery should not force users to reconfigure the whole system (i.e., one should have a smooth recovery process without the need for user involvement).

Secure orphan recovery is accomplished using the security toolbox's key-transport primitive. It involves a two-step process: enrollment and recovery. When an end device first connects to a router, it establishes a secure connection and then executes the orphan enrollment procedure. This procedure involves the end device encrypting its configuration information (e.g., long-term keys and access control lists) and storing this data on the more powerful router device, which has nonvolatile memory. If at some later time the end device becomes an orphan, it will execute the orphan recovery procedure to recover its configuration information from its route (who acts as a secure remote memory service for the device). Note that this allows storage of configuration data securely anywhere on the network and arbitrary replication of secured data, should this be required.

The orphaning steps rely on a unique 128-bit *Orphan Key* stored on each device. For example, this key can be kept in any type of one-time-programmable memory, such as laser-etched memory cells. An RFD already needs to store its unique 64-bit IEEE address, so storing an extra orphan key is an acceptable cost. An end device uses this key to protect the configuration data it stores on the router.

*Assisted association*

An installer accomplishes assisted association (i.e., the linking of two devices at the MAC layer) of an end device and a router by initiating a physical action on each of the devices causing the two devices to associate. For example, the installer can push a button on each of the devices within a limited time. After two devices are associated, cryptographic keys can be established to create a secure and authenticated communication link.

*Remote association*

An installer accomplishes remote association of an end device and a router by storing the end device's address into a device table of the router. For example, a bar code reader could read the address of an end device and load it into the router. Upon power-up, the end device is initially an orphan. The router associates only with orphaned devices that are listed in its device table. Remote association is made secure by using the secure orphaning protocol, which means the router is loaded with configuration data protected with the end device's orphan key (i.e., orphan enrollment).

*Portability*

A network device might be portable, such as a universal remote control. Normally, a remote control might communicate and share keys directly with the television it controls. However, when the remote control moves, or other objects in a room move, this communication path may be disrupted. Communication paths through a network are not always consistent and the security solutions need to account for this. Orphan management can be used to securely implement portability. In this case, a device leaves the domain of one router and enters the domain of second router. If the second router tracks the end device's orphan configuration data, then the end device can act as an orphan and reacquire a link with the second router. Use of a second router is also useful for backup purposes. That is, if the primary router becomes disabled, then the orphaned devices belonging to the disabled router can rejoin the network via the backup router.

*Router linking*

Router linking refers to the process of linking two routers based on a mutually trusted third router. One approach being considered is to have the trusted third router supply a master key to the two devices wishing to link. The two devices could then use this master key to establish their own a link key. This approach requires the availability of a trusted intermediary. Mechanisms where this requirement can be lifted also exist.

*Indirect addressing*

Indirect addressing is an approach for allowing an extremely low-cost end device to send a message through a router to one or more other end devices. The initiating device relies on the router to get the message to the other end devices and is unaware of the address(es) the recipient device(s). Indirect addressing can be secured in one of two ways – either the end devices share a common key with each other or the end devices share a common key with the router. If end devices do not share a key with the router, then the router is only relied upon to deliver the message. It cannot manipulate the message without the end devices noticing.

*Multicast*

In some situations, a device needs to send a message to a number of devices. For example, a light switch may need to control ten lights. It would be inefficient for the light switch to send ten separate messages to each of the lights. Multicast will be secured by using a group key that is shared by a group of devices and maintained by the device that formed the group.

*Code download*

It is often desirable to have the ability to download new software to devices to fix bugs or add new features. The integrity and authenticity of new code needs to be protected to prevent rogue software or viruses from being introduced into the system. Code download facilities do not necessarily need to be part of our architecture, since manufacturers might opt for using a proprietary mechanism for code downloaded to their devices. For example, in the simple devices used in our proposed network, downloaded code will be native machine code (not code in an interpreted language such as Java) that is specific to a particular manufacturer's device. The manufacturer would be responsible for digitally signing this code and his devices would use a pre-installed manufacturer root key (stored in ROM) to verify a signature before enabling downloaded code. Nevertheless, the protocols for signature verification or data authentication can easily be provided in our security design, since these can be implemented by re-using building blocks that are required for, e.g., key establishment or authentic data transfer.

*Service Discovery*

In the case of service discovery, network devices will be able to discover and learn about each other's capabilities. This ability enables easier configuration, for example a light switch can automatically discover all nearby lights that it may want to control. On the security side, however, service discovery poses a privacy and security issue. We do not want burglars to be able to drive through a neighborhood and learn which houses have wireless devices, or how many wireless devices of which type are in a particular home. The solution for secure service discovery is to provide a lock out capability for the service discovery feature. One approach would be to force users to manually override this lockout when they want to add a new device to the network. As an example, when a user presses a button to add a device to the network, service discovery may be temporarily enabled for only a brief duration.

## *3.6 Using the Security Toolbox*

The interface to our security toolbox is given using the notation described in the IEEE Standard 802.2 (1998 edition) and used in the IEEE 802.15.4 specification. Each service is represented as a primitive. A primitive can be one of four generic types:

- **Request:** The request primitive is passed to the security toolbox to initiate a security service.
- **Indication:** The indication primitive is passed from the security toolbox to indicate the occurrence of an internal security event that may be logically related to an external request or an internal security event.
- **Response:** The response primitive is passed to the security toolbox to complete a procedure previously invoked by an indication primitive.

- **Confirm:** The confirm primitive is passed from the security toolbox to convey the results of one or more associated previous service requests.

A typical sequence of request, indication, response, and confirm primitives is illustrated in Figure 10.

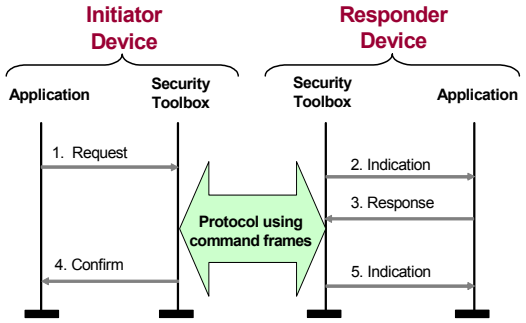


**Figure 10 Message Sequence Chart for Security Primitives**

# 4 Summary and Conclusion

This paper presents a security design for a general-purpose, self-organizing, ad hoc wireless network based on the IEEE 802.15.4 standard. The design supplies message integrity and encryption services, key-establishment protocols (two based on symmetric-key techniques and two based on public-key techniques), and a large set of extended security services, while meeting the low implementation cost, low power consumption, and flexibility requirements of ad hoc wireless networks. Implementation simplicity was enhanced by avoiding repetitive security operations at multiple layers in the communication protocol stack, and by reusing the AES symmetric-key engine to supply multiple security services. ECC was chosen as the public-key cipher due to its small key length and acceptable performance and implementation cost relative to other techniques. The resulting wireless network is sufficiently secure, and offers a wide variety of security services, to meet the requirements of a wide variety of applications. The security architectural design, although described in the context of the IEEE 802.15.4 standard and severe implementation constraints, does not depend hereon and could conceivably be used in any network topology, irrespective of whether these networks are fixed, wireless, or ad hoc wireless.

# References

[1] Frank Stajano, Ross Anderson, "The Resurrecting Duckling: Security Issues in Ad-Hoc Wireless networks," in Bruce Christianson, Bruno Crispo and Mike Roe, eds., Security Protocols, 7th International Workshop Proceedings, Lecture Notes in Computer Science, Springer-Verlag, 1999.

[2] Frank Stajano, "The Resurrecting Duckling: What Next?" in Bruno Crispo and Mike Roe, eds., Proceedings 8th International Workshop on Security Protocols, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Germany, April 2000.

[3] Konrad Wrona, "Distributed Security: Ad Hoc Networks & Beyond," presented at the Ad Hoc Network Security Pampas Workshop, RHUL, London, September 16-17, 2002. Available from http://www.pampas.eu.org/Position_Papers/papers.html.

[4] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks," IEEE Network Magazine, vol. 13, no.6, November/December 1999, pp. 24-30.

[5] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing robust and ubiquitous security support for mobile ad-hoc networks," Proceedings of International Conference on Network Protocols (ICNP), 2001.

[6] S. Capkun, L. Buttyan, and J.-P. Hubaux, “Self-organized public-key management for mobile ad hoc networks,” IEEE Transactions on Mobile Computing, vol. 2, no.1, January-March 2003, pp. 52-64.

[7] M. Steiner, G. Tsudik, and M. Waidner, "Key agreement in Dynamic Peer Groups," IEEE Trans. on Parallel and Distributed Systems, vol. 11, no.8, August 2000, pp. 769-780.

[8] J. Staddon, S. Miner, and M. Franklin, "Self-Healing Key Distribution with Revocation," Proc. IEEE Symp. on Security and Privacy (S&P2002).

[9] Haiyun Luo, Petros Zefros, Jiejun Kong, Songwu Lu, and Lixia Zhang, "Self-securing Ad Hoc Wireless Networks," 7th IEEE Symposium on Computers and Communications (ISCC '02), 2002.

[10] V. Shoup, "Practical Threshold Signatures," Advances in Cryptology, EUROCRYPT '00, pp. 207-220, 2000.

[11] Institute of Electrical and Electronics Engineers, Inc., IEEE Std. 802.15.4-2003, IEEE Standard for Information Technology — Telecommunications and Information Exchange between Systems — Local and Metropolitan Area Networks — Specific Requirements — Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs). New York: IEEE Press. 2003.

[12] FIPS Pub 197, Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, US Department of Commerce/N.I.S.T., Springfield, Virginia, November 26, 2001. Available from http://csrc.nist.gov/.

[13] NIST Pub 800-38A 2001 ED, Recommendation for Block Cipher Modes of Operation – Methods and Techniques, NIST Special Publication 800-38A, 2001 Edition, US Department of Commerce/N.I.S.T., December 2001. Available from http://csrc.nist.gov/.

[14] R. Housley, D. Whiting, and N. Ferguson, “Counter with CBC-MAC (CCM),” submitted to NIST, June 3, 2002. Available from http://csrc.nist.gov/encryption/modes/proposedmodes/.

[15] S. M. Matyas, C. H. Meyer, and J. Oseas, "Generating strong one-way functions with cryptographic algorithm," IBM Tech. Disclosure Bull., vol. 27, no. 10A, 1985, pp. 5658-5659.