

A Secure Framework for Wireless Sensor Networks

B. Barnett and D. Sexton
GE Global Research

Abstract—A large disadvantage to sensor networks in factory environments is the cost of running wire, which ranges from \$40 to \$2000 per foot. The economical advantages of wireless sensor networks is obvious, yet current security frameworks fail to address the complete threat scenarios, and focus on specific threats such as eavesdropping and unauthorized connections. Current proposals ignore the threats of stolen devices, hardware redeployment, and unauthorized hardware reproductions.

This paper describes the design and implementation of a secure and power efficient sensor network, based on off-the-shelf components such as TinyOS software, and the IEEE 802.15.4 compatible Chipcon CC2420 radio. The described protocol conserves battery life by using the hardware-based support for 128-bit AEC-CCM encryption, and all protocol sequences initiated by the sensors.

The authentication system is based on four pair-wise symmetric keys. Two keys, the manufacturer and the customer keys are installed using a serial interface. The third “bootstrap” key is created and distributed over the 802.15.4 network, allowing the sensor to obtain the session key, used for normal operation.

A variation of the Needham-Schroeder protocol is used. Session key updates, multiple session keys, device heartbeats, and software upgrades are supported by the protocol. Six Sigma methodologies were used to evaluate the completeness of the threat reduction, and maximize the completeness of the overall architecture.

This protocol was developed with funding from the Department of Energy, and is being proposed as a standard for manufacturing and factory sites.

Keywords—Secure, wireless, sensor, network

I. INTRODUCTION

WIRELESS sensor networks[1] offers significant advantages in factory environments, as the cost of running wire ranges from \$40 to \$2000 per foot[2]. Readily available sensors such the *tmote Sky*[3] include radios with encryption capability (e.g Chipcon CC2420[4]). However, secure frameworks for sensor networks are lacking robustness, as they tend to focus on a limited number of threats[5]. A complete solution has to consider all threats, and attempt to reduce the risk in as many as possible. It must also

be practical, and flexible to allow deployment in a variety of environments.

In addition, sensor networks present additional implementation challenges. Battery-powered sensors must limit power consumption to be practical. They may also have limited support hardware for encryption such as asymmetric keys. Therefore we devices a framework based on the encryption capabilities built with the Chipcon CC2420 radio.

A. Architecture

Our architecture is based on the requirements for a manufacturing facility or factory. We made several assumptions to simplify the solution. Sensors have limited mobility (i.e. within the factory). There is a central authority (or key distribution center) in a secured room that can be used for device authentication. Our wireless sensors use IEEE 802.15.4 radios. Two types of routers exist – backbone routers that connect the IP network to the 802.15.4 network, and mesh routers that lack IP connectivity. The function of mesh routers is to route 802.15.4 traffic. Sensors may either be mesh routers, or leaf nodes (lacking sophisticated routing capability). As this is a manufacturing facility, we assume off-line authentication is not a requirement: if a central controller is down, new devices shouldn’t be joining the network.

We also assume there are (at least) two authorities or Key Distribution Centers (KDC)– the vendor(s) of the sensors and routers, and the customer who buys the sensors and routers. The identification numbers can be used to partition keys known to the KDC’s.

We assume IP-based networks used in the factory have been secured, as we only address the 802.15.4 wireless communication.

B. Threat Analysis

We considered the risks to both the manufacturer of the sensor network as well as the owner of the facility where the network is deployed. We identified the following threats

- Rogue sensor
- Rogue routers
- Rogue central authority
- Loss of data confidentiality during manufacturing process
- Loss of data confidentiality during installation
- Loss of data confidentiality during device start-up
- Loss of data confidentiality during normal operations
- Loss of data confidentiality during software upgrade

- Data integrity and authenticity
- Man-in-the-middle or replay attacks
- Brute force attack against sensor/router
- Denial of service attack on sensor/router
- Insider attack at vendor’s site
- Insider attack at customer’s site
- Customer’s sensor/router physically attacked on site while connected to the network
- Customer’s sensors/routers stolen and attacked off-site
- Vendor’s sensor/router attacked off-site for reverse engineering
- Vendor’s sensor/router modified off-site and sold to customer
- Sensors/routers stolen from vendor
- Device Cloning – or unauthorized reproductions
- Hardware Redeployment of sensor/router – where the buyer of a used sensor obtains confidential information from the former owner.

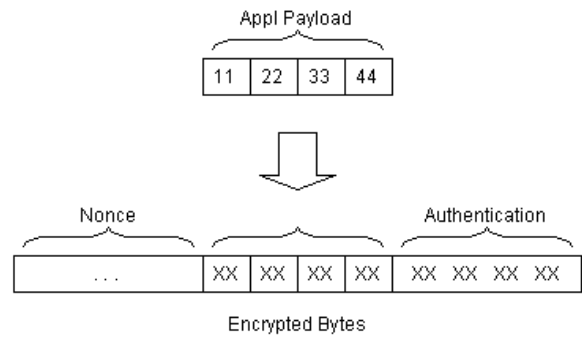


Figure 2 - Data-layer encryption

The generation of the nonce value can be the same source as the MAC-layer nonce. That is, the same monotonically increasing value can be used. This simplifies the detection of replay attacks by the sensor.

We categorized the on-site and off-site attacks as different, because the difference in detection and response. Also note that reverse engineering, devices with back doors, and physical attacks are considered as threats as well.

No secure framework can eliminate all threats. However, all of the threats must be considered to evaluate the effectiveness of the approach.

II. SECURITY FRAMEWORK

A. Encryption support in 802.15.4 radio

802.15.4[6] packets support several MAC (Media Access Control)-layer encryption and authentication schemes[7]. In our prototype, we used the hardware-based AES-CCM-128 for both data-layer and MAC encryption and authentication. The recommendations made in ZigBee[8] for the nonce and use of CCM* instead of CCM should be followed. Our primary goal was to obtain the best available security with existing and readily available encryption hardware.

We used TinyOS as a development environment. Our packet format is as follows:

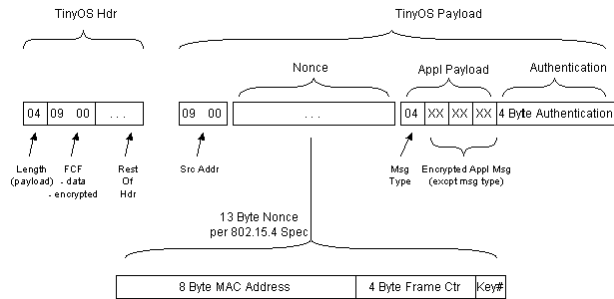


Figure 1 - MAC Packet Format

TinyOS automatically updates the data sequence number in header. In addition, we use a monotonically increasing number for the 4-byte Frame Counter.

Data-layer encryption has the following structure.

B. Generalized Key Installation Mechanism

We use the following terminology: Sensors S with unique identifier I_S communicate to authorities A to obtain keys K . Encrypted messages are indicated by $\{\dots\}K$.

Authentication is based on paired symmetric keys, where only two devices use the key to authenticate each other. The generalized key installation is based on principles used in Needham-Schroeder[9], and Kerberos[10], shown in these three steps.

- (1) $S \rightarrow A: I_S, N_1$
- (2) $A \rightarrow S: \{I_S, N_1, N_2, K_{N+1}\} K_N$
- (3) $S \rightarrow A: N_2$

Key K_N is therefore used to install key K_{N+1} . The sensor verifies the key validity by checking the identification number and nonce within the encrypted response matches. The nonces N_1, N_2 are used to prevent replay attacks.

The primary threat in this approach is knowledge of K_N beforehand, or capturing traffic and obtaining the key afterwards (forward security). Nonce predictability is an issue for the authority A in step 3. A cryptographically strong random number generator (RNG) is suggested to prevent this. The nonce for the sensors is a potential problem. Trusted timestamps and RNG’s require additional resources that may not be available in a sensor; we therefore assume as a worst case that sensors use a monotonically increasing number, and authentication devices that validate these numbers remember the last number used for each sensor.

In general, lower number keys have greater longevity, are more valuable in controlling assets, and used less often. As shall be described, higher number keys are used to provide localized scope for authentication. The responsibilities on the sensor are minimized, as it doesn’t need to know anything about the authority that generated the key. Therefore keys can be passed onto additional authorities.

Relays and third parties can be used to install keys, if one is willing to accept the additional risk. Nonce N_2 is used to confirm the device received the proper key.

Nonce lifetime can be sufficient to allow for long-latency authentication, which allows a form of off-line authentication

described later.

We also use “ \rightarrow ” to indicate the transmission of a packet without MAC encryption, and “ \Rightarrow ” to indicate MAC encryption as a visual aid.

There are advantages to making this key installation mechanism generalized. We originally proposed four keys for the framework, but two additional keys allow greater flexibility, as described below.

C. Vendor key installation

We envision a sensor manufacturing facility produces nearly identical devices, installing a common key K_0 into the firmware, while installing a unique identification into the device. A key installation mechanism installs a unique key K_I into the device. As K_0 is known, this must be a private and secured communication channel:

- (4) $S \rightarrow A: I_S, N_1$
- (5) $A \rightarrow S: \{I_S, N_1, N_2, K_I\} K_0$
- (6) $S \rightarrow A: N_2$

This key can, if the vendor desires, be used to install a secondary vendor key K_2 .

Key K_2 can be constructed to expire after a period of time.

The device identification and matching keys must be stored in the vendor’s KDC, which must obviously be in a secured room.

D. Customer Key Installation

When the customer received the sensor, and obtains the identification, it obtains a key K_V from the vendor. Depending upon the implementation, this can be K_I or K_2 . There are several mechanisms that can be used to obtain the key, such as a CD-ROM, a web server, tear-off labels, etc. The vendor can revoke keys based on I_S in the case of stolen or cloned devices by refusing to tell the customer the key. The vendor can verify the identity of the device by examining N_2 . If K_V is K_2 , the key can be expired to force the customer to refresh the key. If K_V is K_I , once that key is obtained the customer need not use the vendor services to enable the sensor in the future, and the vendor cannot revoke the key in the customer’s possession.

The customer then installs the unique-per-device site key K_S .

- (7) $S \rightarrow A: I_S, N_1$
- (8) $A \rightarrow S: \{I_S, N_1, N_2, K_S\} K_V$
- (9) $S \rightarrow A: N_2$

And can optionally install one or more end-to-end authenticity keys, such as K_A , using

- (10) $S \rightarrow A: I_S, N_3$
- (11) $A \rightarrow S: \{I_S, N_3, N_4, K_A\} K_S$
- (12) $S \rightarrow A: I_S, N_4$

Steps (7), (8) and (9) can be performed for multiple devices in batches, if desired, provided that the sensor doesn’t time out. A partial form of off-line authentication could be done with a hand-held as a proxy with partial connectivity. It must

first collect and relay all N_1 values, relay the responses, and collect all of the N_2 values and forward them to the KDC so it can verify the keys were installed.

We also considered key installation schemes where a third party installs the network, has access to the KDC, and hands it over to the customer. In this case, the customer must refresh all of the K_V keys, and then the K_S keys over the network to prevent the installer from gaining access. This requires at least two vendor keys, as K_2 must be revoked.

III. WIRELESS PROTOCOL SEQUENCES

Battery-conserving sensors may choose to “sleep” – with their radios disabled. Therefore in our design, the sensor initiates all wireless protocol sequences described below. Protocol sequences typically end with a status packet from the router to a non-routing sensor, which identifies pending sequences. Essentially this allows non-routing sensors to sleep for extended periods with the radio disabled. Once they wake up, and transmit the data, the status packet tells them if they have any additional tasks (key update, software update, etc.) before returning to a sleep state.

A. Joining the network

A sensor joining the network for the first time needs to locate and verify the identity of the nearest router. We assume the router has already been authenticated for the network, and has its own key K_{Router} known to the site authority, which is shared with KDC. The router has a unique identification number, like the sensor, here described as I_R .

We use a two-phase sequence to obtain the MAC key similar to the SPINS protocol[11]. The first sequence obtains the bootstrap key $K_{Bootstrap}$.

- (13) $S \rightarrow Broadcast:$
- (14) $R \rightarrow S: I_R$
- (15) $S \rightarrow R: I_S, \{I_S, I_R, N_1\} K_S$
- (16) $R \rightarrow A: \{I_R, I_S, N_2, \{I_S, I_R, N_1\} K_S\} K_{Router}$
- (17) $A \rightarrow R: \{K_{Bootstrap}, I_R, I_S, N_2\} K_{Router} + \{K_{Bootstrap}, I_S, I_R, N_1\} K_S$
- (18) $R \rightarrow S: \{K_{Bootstrap}, I_S, I_R, N_1\} K_S$

This sequence allows the site authority to create and install a new bootstrap key that allows the router and sensor to authenticate each other. Note that the message in (15) is opaque to the router, which forwards it to the site authority for authentication in (16). The site authority returns two encrypted messages in (17), and the router authenticates and forwards one of these to the sensor in (18).

Once the bootstrap key is obtained, the sensor is able to get the MAC key, K_{MAC} , along with an 8-bit key identification number $K_{\#}$.

- (19) $S \rightarrow R: \{I_S, I_R, N\} K_{Bootstrap}$
- (20) $R \rightarrow S: \{K_{MAC}, K_{\#}, I_S, I_R, N\} K_{Bootstrap}$
- (21) $S \Rightarrow R: \{Acknowledge\} K_{MAC}$
- (22) $R \Rightarrow S: \{Status\} K_{MAC}$

Steps 19-21 can be used to re-join a network if K_{MAC}

expires without burdening the KDC.

As $K_{\#}$ is 8 bits wide, the 802.15.4 radio supports up to 256 keys for MAC encryption. These keys may be shared for all devices, and different keys can be used for broadcast and multicast. Note that sequences (21) and (22) use MAC encryption key K_{MAC} .

This mechanism places a greater burden on the router than other proposals, but it allows greater control (each K_{MAC} could be unique per router/sensor pair, or shared within all nodes connected to a router). It also provides greater scalability, as will be discussed below. Also note that routing sensors join the network like non-routing sensors (K_{Router} could be K_S from the router's perspective).

B. MAC Key Management

The 802.15.4 packet specifications support 256 different keys and the key number $K_{\#}$ corresponds $Key\#$ in Figure 1. The CC2420 only supports 2 keys in hardware, but we believe the sensor can examine the header of the packet, determine the key number, install the matching key into the appropriate memory register, and then decrypt the rest of the packet in hardware.

The status word is used to indicate pending actions for the sensor. This may include the number of keys pending, number of valid keys, and time until the current key expires. Therefore the sensor can repeat sequences 19-22 to obtain additional keys.

The router keeps track of the keys known to each sensor. This allows the router to change keys once it knows that all of the sensors have obtained the new key. It also allows the router to partition the knowledge of the subordinate sensors, allowing it to quickly exclude a compromised sensor.

At the end of each sequence, the sensor will examine the status packet, and decide if it needs to perform additional actions. One of these is to ask for a new MAC key. This sequence is as follows:

- (23) $S \rightarrow R: \{I_S, I_R, N\} K_{Bootstrap}$
- (24) $R \rightarrow S: \{K_{MAC}, K_{\#}, I_S, I_R, N\} K_{Bootstrap}$
- (25) $S \Rightarrow R: \{Acknowledge\} K_{MAC}$
- (26) $R \Rightarrow S: \{Status\} K_{MAC}$

Sequences (23) and (23) could also use MAC encryption as well for increased security.

The router can pre-load several keys in advance, and keep track of which sensors know which key numbers. This allows the router to smoothly switch to a new MAC key by selecting one known to all devices. It can purposely exclude or isolate devices by category if it needs to. Key changes can be synchronized by clock as well, with the status telling the sensor the time till the next key change. As the 8-bit key number is disclosed in the packet header, a device can potentially switch between keys on a packet-by-packet basis, allowing a smooth transition from one key to the next without dropping MAC-layer encryption.

If a device discovers it no longer has a valid MAC key, it can fall back to (19) or even (13) to obtain the key. We believe that allowing this to happen at the router layer offloads

the KDC, and enhances scalability.

C. Alarm

While the network is in operation, devices can be physically attacked. Preventing these attacks is difficult [12][13]. We added an alarm mechanism, assuming the device might have some way to detect intrusion attacks, either physical or over the network (i.e. brute force, replay, etc.):

- (27) $S \Rightarrow R: \{Alarm\} K_{MAC}$
- (28) $R \Rightarrow S: \{Status\} K_{MAC}$

A device that detects physical attacks and sends an alert can allow the router and site authority to react in a series of escalations; first by changing any shared K_{MAC} . The KDC or router can then revoke the $K_{Bootstrap}$ key of any compromised sensor and temporarily prevent any new key from being issued. Finally the sensor's K_S key can be revoked if the device is stolen or redeployed.

D. Heartbeat

Another concern is an attacker who can physically compromise a device without detection, while the device is connected to the network, and obtain shared K_{MAC} keys.

We use a heartbeat function to detect unavailable or missing devices.

- (29) $S \Rightarrow R: \{I_S, I_R, N^1\} K_{MAC}$
- (30) $R \Rightarrow S: \{\{I_S, I_R, N^1, N^2\} K_{Bootstrap}\} K_{MAC}$
- (31) $S \Rightarrow R: \{I_S, I_R, N^2\} K_{MAC}$
- (32) $R \Rightarrow S: \{Status\} K_{MAC}$

This allows the sensor and router to authenticate each other. Routers can summarize the information and pass it up to the site authority.

The response to a missing device can escalate in time, as mentioned earlier. Keys can be changed, temporarily prevented from re-issuing, and finally revoked if the decision is made that the device is lost. This functionality could be implemented by a central authority, but by localizing this to the router increases scalability.

Offloading the responsibilities of a central KDC onto a router, with tasks such as network re-joining, key updates and heartbeats, increases the scalability of the network.

E. End-to-End Authentication

Up to this point, the framework provides link-to-link authentication. However, a trusted but compromised router could modify data en route.

In addition, some may wish to have the ability for intrusion detection devices to log and inspect packet contents, and optionally decrypt data.

To allow this, we propose end-to-end authenticity key K_A as installed in 10-12. A sensor could use this to sign an/or encrypt data in addition to MAC-encryption. Any device that knew this key could authenticate/decrypt the data, but be unable to join the network or obtain the MAC-layer key used for that link. Routers could forward these packets to logging devices.

The key installation protocol could be used over-the-air to update/replace the end-to-end authenticity key, while giving the old key to intrusion detection devices. These devices should be prevented from capturing the packets where keys are updated.

F. Software Update

TinyOS provides a way to update software with the Deluge extension. However, the existing mechanism provides no way to verify the authenticity of the software.

We propose that the firmware be transmitted using MAC encryption, and hash of the firmware H , along with the revision identification I_{REV} , be signed by K_S :

$$(33) R \Rightarrow S: \{\{H, I_{REV}\}K_S\}K_{MAC}$$

IV. KEY REVOCATION

In general, the vendor creates K_V , and the customer site authority creates the other keys. Optionally the router may create K_{MAC} for efficiency reasons.

The keys can be revoked under conditions listed in Table 1. Note that if a key is revoked, the key in the table above it must be used to obtain another key.

Table 1 - Key Revocation Conditions

Key	Revocation Condition
K_V	<ul style="list-style-type: none"> ● Unauthorized reproduction of device ● Large shipment of devices stolen ● Key expired (if using two vendor keys)
K_S	<ul style="list-style-type: none"> ● Device stolen, missing, or compromised ● Device sold or redeployed
$K_{Bootstrap}$	<ul style="list-style-type: none"> ● Key expired ● Device under attack
K_{MAC}	<ul style="list-style-type: none"> ● Key expired/updated ● Unauthorized device on network
K_A	<ul style="list-style-type: none"> ● Device expired/updated ● Key exported for analysis

The vendor could also revoke individual keys if desired, in high-security situations. A more practical approach is to simply let the vendor key expire, which would limit the deployment of stolen devices.

V. THREAT ANALYSIS

We used a 6-Sigma methodology[14] to evaluate the framework. It is a Failure Mode Effect Analysis (FMEA), adapted for risk assessments. Three values are multiplied together to calculate a Risk Prioritization Number (RPN): Likelihood, Severity and Detectability. Each value ranges from 1 to 10, with higher numbers being worse. In this

example, the values are shown in Tables 2-4. The group must first reach consensus of the meanings of the values.

Each threat is subjectively compared to other threats, as long as all parties agree to the values. Individual values for single threats aren't significant. Instead, threats are given values when compared to other threats. The advantage of this method is not in absolute values, but in determining which risks are more frequent, detectable, and dangerous than other threats. This technique is practical, as it provides a simple methodology to consider all threats, and workable, as it makes no attempts to be universally objective. Sample values are given in tables 2, 3 and 4. The vendor and the customer will normally have separate tables and values; we combined them for this paper to simplify the analysis.

We have filled in the values of a theoretical factory that is using wireless sensors. We subjectively compare an unsecured wireless framework to the framework described in this paper. The results are shown in Table 5.

Note that for an unsecured system, using a merged vendor/customer view, the greatest threats are: (1) device cloning, (2) loss of data confidentiality during normal operations, (3) stolen/black market devices, and (4) data integrity. Using the proposed framework, note that nearly every threat is reduced, and the total risk is reduced from 4737 to 1948 – a reduction of 58.88%. The average threat is reduced from 225.57 to 92.76 in this example.

With the new framework, some individual values may increase. In particular, the severity of loss of data confidentiality during manufacturing increases, as the keys may be compromised. Also note that some threats are not addressed in the framework – such as the risk of an insider at the customer's site.

VI. DISCUSSION OF IMPLEMENTATION

The Chipcon device efficiently handles encryption in hardware. Encrypting packets at the link level using CCM-128 increases transmission time approximately 500 μ secs in some of the tests we have tried.

In our prototype, the source address and packet type were not encrypted to simplify debugging. However, they should be encrypted for increased security, and the cost of increased debugging complexity. In particular, if the attacker cannot distinguish packets that accomplish a key update, they would need to retain more packets in the event that an older key is obtained.

We realized that the Chipcon CC2420 had a problem receiving both unencrypted and encrypted packets. If we expect the next packet to be plaintext, and an encrypted packet arrives, the hardware cannot decode the packet. To address this, we always transmit encrypted packets. We emulate unencrypted packets by encrypting these packets using a key known to all devices. We don't consider this a security feature, but an implementation detail. Future radios may not have this limitation.

We implemented a prototype based on a preliminary version of this framework, with the 4 essential keys described.

This paper describes a more flexible and adaptive approach, based on our experience, and based on requirement discussions with several vendors.

This paper recommends several implementation details. However, in the case where cost and convenience is more important than security, certain security protection mechanisms can be eliminated. We propose that both MAC layer and data-layer encryption be used for maximum security. Therefore the data is encrypted twice. A single encryption step could be used to extend battery life. In the case of low-cost and low-security sensors, a single vendor key could be shared across several devices. This provides no protection from cloned devices, but that's a decision the vendor can make.

Different sites can determine the number of K_{MAC} keys to use. One site can use a single shared key, a second can use one for each router, and a third can use unique keys for every link.

The number of keys necessary to remember is proportional to the importance of a device in the infrastructure. Assuming the 4-key mechanism (K_V , K_S , $K_{Bootstrap}$, K_{MAC}), an end-node sensor would need to know $2+2N$ keys, where N is the number of mesh routers it talks to. A mesh router, with N peer routers, and M leaf nodes, would need to know up to $2+2N+2M$ keys. Because of the limitations in the 802.15.4 frame, each router is limited to 256 different MAC-layer keys. The KDC has the greatest burden, by needing to know all of the keys. The worst case would be unique K_{MAC} keys, with every device able to communicate with every other device. However, in practical terms, this is rarely needed.

VII. CONCLUSIONS

We feel that for environments found in factories, this framework is practical, flexible, and scalable. The encryption technology is based on pairwise symmetric keys, allowing low-cost deployment in embedded systems. We built a prototype using this technology, to validate the concepts.

Using two KDC's allows vendors and customers to protect their assets, without burdening the device with knowledge about different authorities. Three or more KDC's are possible as well. The use of multiple keys in a one-way chain provides flexibility in deployment, installation and revocation of keys. We can conceive a tree-structure of chains for asset control.

The framework provides multiple ways to revoke and redeploy devices without compromising a network. It also allows for quick detection and key revocation for devices under active attacks. Complex situations, such as allowing hand-offs to third parties once the keys have been established, are also allowed, as well as protection from device cloning.

Use of 6-Sigma methodology allowed us to consider a comprehensive threat assessment, as well as provided a mechanism to subjectively evaluate the effectiveness of the design.

We believe this is suitable for a wide range of requirements, and consider it suitable as a standard for wireless automation,

as promoted by ISA's SP100 working group.[15] Other applications may also find this framework useful.

Table 2 - Likelihood Values

Probability of Security failure	Proposed Failure Rate	Value
Very High: Failure is almost inevitable	Several times a day	10
	Daily	9
High: Repeated failures	Weekly	8
		7
Moderate: Occasional failures	Monthly	6
	Yearly	5
		4
Low: Relatively few failures	Once every 10 years	3
	Once every 100 Years	2
Remote: Failure is unlikely	Once every 1000 years	1

Table 4 - Severity Values

Effect	Proposed Severity of Effect	Value
Hazardous without warning	Major loss of life. Permanent financial impact	10
Hazardous with warning	Widespread abuse. Crisis. Large financial impact	9
Very High	Loss of large customers/accounts	8
High	Significant loss of value	7
Moderate	Notable loss of value	6
Low	Minor loss of value	5
Very Low	Very minor loss of value	4
Minor	Loss of single customer	3
Very Minor	Inconvenient	2
None	No effect	1

Table 3- Detection Values

Detection	Likelihood of Detection by Design Control	Value
Absolute Uncertainty	Design control cannot detect potential cause/mechanism and subsequent failure mode	10
Very Remote	Very remote chance	9
Remote	Remote chance	8
Very Low	Very low chance	7
Low	Low chance	6
Moderate	Moderate chance	5
Moderately High	Low chance	4
High	High chance	3
Very High	Very high chance	2
Almost Certain	Design control will detect failure	1

Table 5 - Failure Mode Effect Analysis

Threat	Current Architecture				Proposed Architecture				Relative Reduction in Risk (%)
	L	D	S	RPN	L	D	S	RPN	
Rogue sensor	6	5	7	210	6	1	7	42	80.0
Rogue routers	6	5	7	210	6	1	7	42	80.0
Rogue central authority	3	2	8	48	3	1	8	24	50.0
Loss of data confidentiality during manufacturing process	6	8	3	144	3	7	5	105	27.1
Loss of data confidentiality during installation	6	8	3	144	3	8	3	72	50.0
Loss of data confidentiality during device start-up	6	8	3	144	2	8	3	48	66.7
Loss of data confidentiality during normal operations	9	8	8	576	2	8	8	128	77.8
Loss of data confidentiality during software upgrade	4	8	6	192	2	8	6	96	50.0
Data integrity and authenticity	9	4	9	324	2	2	9	36	88.9
Man-in-the-middle attacks	5	6	3	90	2	2	3	12	86.7
Brute force attack against sensor/router	5	6	6	180	5	5	6	150	16.7
Denial of service attack on sensor/router	6	5	6	180	6	3	6	108	40.0
Insider attack at vendor's site	4	7	3	84	4	7	2	56	33.3
Insider attack at customer's site	5	7	6	210	5	7	6	210	0.0
Customer's sensor/router physically attacked on site while connected to the network	4	4	6	96	4	3	3	36	62.5
Customer's sensors/routers stolen and attacked off-site	6	5	4	120	6	3	3	54	55.0
Vendor's sensor/router attacked off-site for reverse engineering	7	10	3	210	9	10	2	180	14.3
Vendor's sensor/router modified off-site and sold to customer	4	8	9	288	3	8	9	216	25.0
Sensors/routers stolen from vendor	8	9	7	504	8	9	3	216	57.1
Device Cloning – or unauthorized reproductions	8	9	9	648	8	3	3	72	88.9
Hardware Redeployment of sensor/router	5	9	3	135	5	3	3	45	66.7
Total				4737				1948	58.9
Max Value				21000				21000	
Average Threat Value				225.57				92.76	
Overall				22.56%				9.28%	

REFERENCES

ACKNOWLEDGMENT

We would like to acknowledge the assistance of Rene Struik and Darryl Parisien of Certicom for their assistance. Ron Olson implemented the framework on the TinyOS platform, and Ping Liu also provided assistance in the project management.

- [1] Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, 2002. Wireless sensor net- Rensselaer Polytechnic Institute
- [2] Wireless Sensors: Where Are the 800-Pound Gorillas?’, Frost & Sullivan, 2 Jan 2003 | Industrial Automation
- [3] <http://www.moteiv.com/>
- [4] Chipcon CC2420 Data Sheet http://www.chipcon.com/files/CC2420_Data_Sheet_1_4.pdf
- [5] S. A. Camtepe and B.. Yener “Key Distribution Mechanisms for Wireless Sensor Networks: a Survey”

- [6] IEEE 802 Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANS). Available: <http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf>
- [7] N. Sastry, D. Wagner, "Security Considerations for IEEE 802.15.4 Networks" in *Proceedings of the 2004 ACM workshop on Wireless security*, Philadelphia, PA, USA 2004
- [8] Zigbee alliance. <http://www.zigbee.org>
- [9] R. Needham and M. Schroeder. "Using Encryption for Authentication in Large Networks of Computers". *Communications of the ACM*, 21(12), December 1978.
- [10] J.G. Steiner, C. Neuman, and J. I. Schiller. Kerberos: "An Authentication Service for Open Network Systems". jan 1988.
- [11] A. Perrig., R. Szewczyk,., V. Wen, D. Culler, D. Tygar, "SPINS: Security Protocols for Sensor Networks" *Proceedings of Seventh Annual International Conference on Mobile Computing and Networks MOBICOM 2001*, July 2001.
- [12] R. Anderson, M. Kuhn, "Tamper Resistance a Cautionary Note" *Proceedings of the Second Usenix Workshop on Electronic Commerce* (1996)
- [13] Alexander Becher, Zinaida Benenson, Maximillian Dornseif: *Tampering with Motes: Real-World Attacks on Sensor Networks*. 3rd International Conference on Security in Pervasive Computing (SPC), April 2006, York, UK. <http://pi1.informatik.uni-mannheim.de/publications/pdf/tampering-with-motes-real-world-attacks-on-wireless-sensor-networks>
- [14] B. Barnett "A Practical Top-down Threat Assessment using 6-Sigma Methodology" GE Technical Report 2004
- [15] ISA-SP100, Wireless Systems for Automation www.isa.org/isasp100/