Security Protocols for Sensor Networks Using Protocol Architecture

NAGA GOPI KARANKI#1, Dr.A SrinivasaaRao#2.

COMPUTER NETWORKS & SECURITY [M.TECH]

K L University, VADDESWARAM, GUNTUR

Email:karanki.nagagopi88@gmail.com

Prof of K L University

Email:dsrinivas@kluniversity.in

Abstract: **Wireless sensor networks will be widely deployed in the near future. While much research has focused on making these networks feasible and useful, security has received little attention. We present a suite of security protocols optimized for sensor networks: SPINS. SPINS has two secure building blocks: SNEP and µTESLA. SNEP includes: data confidentiality, two-party dataauthentication, and evidence of data freshness. µTESLA provides authenticated broadcast for severely resource-constrained environments. We implemented the above protocols, and show that they are practical even on minimal hardware: the performance of the protocol suite easily matches the data rate of our network. Additionally, we demonstrate that the suite can be used for building higher level protocols.**

Keywords: **secure communication protocols, sensor networks, mobile ad hoc networks, MANET, authentication of wireless communication, secrecy and confidentiality, cryptography**.

1. Introduction

We envision a future where thousands to millions of small sensors form self-organizing wireless networks. How can we provide security for these sensor networks? Security is not easy; compared with conventional desktop computers, severe challenges exist – these sensors will have limited processing power, storage, bandwidth, and energy. We need to surmount these challenges, because security is so important. Sensor networks will expand to fill all aspects of our lives. Here are some typical applications:

a) *Emergency Response Information*: sensor networks will collect information about the status of buildings, people, and transportation pathways. Sensor information must be collected and passed on in meaningful, secure ways to emergency response personnel.

b) *Energy Management*: in 2001 power blackouts plagued California. Energy distribution will be better managed when we begin to use remote sensors. For example, the power load that can be carried on an electrical line depends on ambient temperature and the immediate temperature on the wire. If these were monitored by remote sensors and the remote sensors received information about desired load and current load, it would be possible to distribute load better. This would avoid circumstances where Californians cannot receive electricity while surplus electricity exists in other parts of the country.

c) *Medical Monitoring*: we envision a future where individuals with some types of medical conditions receive constant monitoringthroughsensorsthatmonitorhealth conditions. For some types of medical conditions, remote sensors may apply remedies (such as instant release of emergency medication to the bloodstream).

i. Logistics and inventory management: commerce in America is based on moving goods, including commodities from locations where surpluses exist to locations where Needs exist. Using remote sensors can substantially improve these mechanisms. These mechanisms will vary in scale – ranging from worldwide distribution of goods through transportation and pipeline networks to inventory management within a single retail store.

ii. Battlefield management: remote sensors can help eliminate some of the confusion associated with combat. They can allow accurate collection of information about current battlefield conditions as well as giving appropriate information to soldiers, weapons, andvehiclesin the battlefield.At UC Berkeley, we think these systems are important, and we are starting a major initiative to explore the use of wireless sensor networks. (More information on this new initiative, CITRIS, can be found at www.citris.berkeley.edu.) Serious security and privacy questions arise if third parties can read or tamper with sensor data. We envision wireless sensor networks being widely used – including for emergency and life-critical systems – and here the questions of security are foremost. This article presents a set of Security Protocols for Sensor Networks, SPINS. The chief contributions of this article are:

iii. Exploring the challenges for security in sensor networks.

iv. Designing and developing µTESLA (the "micro" version of TESLA), providing authenticated streaming broadcast.

v. Designing and developing SNEP (Secure Network Encryption Protocol) providing data confidentiality, two- party data authentication, and data freshness, with low overhead.

vi. Designing and developing an authenticated routing protocol using our building blocks.

1.1. *Sensor Hardware*: At UC Berkeley, we are building prototype networks of small sensor devices under the Smart Dust program, one of the components of CITRIS. We have deployed these in one of

Characteristics of prototype Smart Dust nodes:

CPU	8-bi 4 MHz
Storage	8 Kbytes
Instruction flash	512 bytes
RAM	512 bytes
EEPROM Communication	916 MHz radio
Bandwidth	10 Kbps
Operating system	Tinos
OS code space	3500 bytes
Available code space	4500 bytes

The operating system is particularly interesting for these devices. We use Tangos. This small, event-driven operating system consumes almost half of 8 Kbytes of instruction flash memory, leaving just 4500 bytes for security and the application.

It is hard to imagine how significantly more powerful devices could be used without consuming large amounts of power. The energy source on our devices is a small battery, so we are stuck with relatively limited computational devices. Wireless communication is the most energy-consuming function performedbythese devices,so we need to minimize communications overhead. The limited energy supplies create tensions for security: on the one hand, security needs to limit its consumption of processor power; on the other hand, limited power supply limits the lifetime of keys.

2. System Assumptions

Before we outline the security requirements and present our security infrastructure, we need to define the system architecture and the trust requirements. The goal of this work is to propose a general security infrastructure that is applicable to a variety of sensor networks.

2.1 Communication Architecture: Generally, the sensor nodes communicate over a wireless net- work, so broadcast is the fundamental communication primitive. The baseline protocols account for this property: on one hand they affect the trust assumptions, and on the other they minimize energy usage.

We do have an advantage with sensor networks, because most communication involves the base station and is not between two local nodes. The communication patterns within our network fall into three categories:

• Node to base station communication, e.g., sensor readings.

• Base station to node communication, e.g., specific requests.

• Base station to all nodes, e.g., routing beacons, queries or reprogramming of the entire network.

Our security goal is to address these communication patterns, though we also show how to adapt our baseline protocols to other communication patterns, i.e. node to node or node broadcast.

2.2 Design Guidelines: With the limited computation resources available on our plat- form, we cannot afford use asymmetric cryptography and so we use symmetric cryptographic primitives to construct the SPINS protocols. Due to the limited program store, we construct all cryptographic out of a single block cipher for code reuse. To reduce communication overhead we exploit common state between the communicating parties. Requirements for sensor network security. This section formalizes the security properties required by sensor networks, and shows how they are directly applicable in a typical sensor network.

3. Requirements for Sensor Network Security

This section formalizes the security properties required by sensor networks, and shows how they are directly applicable in a typical sensor network.

3.1 *Data Confidentiality*: A sensor network should not leak sensor readings to neigh- boring networks. In many applications (e.g., key distribution) nodes communicate highly sensitive data. The standard approach for keeping sensitive data secret is to encrypt the data with a secret key that only intended receivers possess, hence achieving confidentiality. Given the observed communication patterns, we set up secure channels between nodes and base stations and later bootstrap other secure channels as necessary.

3.2 Data Authentication: Message authentication is important for many applications in sensor networks (including administrative tasks such as net- work reprogramming or controlling sensor node duty cycle). Since an adversary can easily inject messages, the receiver needs to ensure that data used in any decision-making process originates from a trusted source. Informally, data authentication allows a receiver to verify that the data really was sent by the claimed sender. Informally, data authentication allows a receiver to verify that the data really was sent by the claimed sender.

3.3 Data Integrity: In communication, data integrity ensures the receiver that the received data is not altered in transit by an adversary. In SPINS, we achieve data integrity through data authentication, which is a stronger property.

3.4 Data Freshness: Sensor networks send measurements over time, so it is not enough to guarantee confidentiality and authentication; we also must ensure each message is fresh. Informally, data freshness implies that the data is recent, and it ensures that no adversary replayed old messages. We identify two types of freshness: weak freshness, which provides partial message ordering, but carries no delay information, and strong freshness, which

provides a total order on a request–response pair, and allows for delay estimation. Weak freshness is useful for sensor measurements, while strong freshness is useful for time synchronization within the network.

4. Notation

We use the following notation to describe security protocols and cryptographic operations in this article:

i. A, B are principals, such as communicating nodes.
ii. NA is a nonce generated by A.
iii. XAB denotes the master secret (symmetric) key which is shared between A and B. No direction information is stored in this key, so we have XAB = XBA.
iv. KAB and KBA denote the secret encryption keys shared between A and B. A and B derive the encryption key from the master secret key XAB based on the direction of the communication: KAB = FXAB (1) and KBA=FXAB (3), where F is a Pseudo-Random Function (PRF)
v. KAB and KBA denote the secret MAC keys shared between A and B. A and B derive the encryption key from the master secret key XAB based on the direction of the communication: KAB = FXAB (2) and KBA = FXAB (4), where F is a pseudo-random function.
vi. {M}K AB is the encryption of message M with the encryption key KAB.
vii. {M} K AB, denotes the encryption of message M, with key KAB, and the initialization vector IV which is used in encryption modes such as cipher-block chaining (CBC), output feedback mode (OFB), or counter mode (CTR)
viii. MAC (KAB, M) denotes the computation of the message authentication code (MAC) of message M, with MAC key KAB.

By a secure channel, we mean a channel that offers confidentiality, data authentication, integrity, and freshness.

5. Spins Security Building Blocks

To achieve the security requirements we established in we design two security building blocks: SNEP and µTESLA. SNEP provides data confidentiality, two-party data authentication, integrity, and freshness. µTESLA provides authentication for data broadcast. We bootstrap the security for both mechanisms with a shared secret key between each node and the base station. We demonstrate in section 8 how we can extend the trust to node-to-node inter- actions from the node-to-base-station trust.

5.1 SNEP: Data confidentiality, Authentication, Integrity, and Freshness: SNEP provides a number of unique advantages. First, it has low communication overhead; it only adds 8 bytes per message. Second, like many cryptographic protocols it uses a counter, but we avoid transmitting the counter value by keeping state at both end points. Third, SNEP achieves semantic security; a strong security property which prevents eaves droppers from inferring the message content from the encrypted message.Finally, the same simple and efficient protocolalso givesus data authentication, replay protection, and weak message freshness.

Data confidentiality is one of the most basic security primitives and it is used in almost every security protocol. A simple form of confidentiality can be achieved through encryption, but pure encryption is not sufficient. Another important security property is semantic security, which ensures that an eavesdropper has no information about the plaintext, even if it sees multiple encryptions of the same plaintext For example, even if an attacker has an encryption of a 0 bit and an encryption of a 1 bit, it will not help it distinguish whether a new encryption is an encryption of 0 or 1. A basic technique to achieve this is randomization: Before encrypting the message with a chaining encryption functions the sender precedes the message with a random bit string. This prevents the attacker from inferring the plaintext of encrypted messages if it knows plaintext ciphertext pairs encrypted with the same key.

A good security design practice is not to reuse the same cryptographic key for different cryptographic primitives; this prevents any potential interaction between the primitives that might introduce a weakness. Therefore we derive independent keys for our encryption and MAC operations. The two communicating parties A and B share a master secret key XAB, and they derive independent keys using the pseudo- random function F: encryption keys KAB = FX (1) and KBA = FX(3) for each direction of communication, and Mac keysKAB = FX(2) and KBA = FX(4) for each direction of communication. Section 6 gives more details on key derivation.

The combinations of these mechanisms form our Sensor Network Encryption Protocol SNEP. The encrypted data has the following format: E = {D} K, C., whereDis the data, the encryption key is K, and the counter is C.TheMACis M=MAC (K, C‖E). The complete message that A sends to B is

A → B: {D} KAB, CA, MACK'ABCA ‖ {D} KAB, CA.

5.2 Co*unter Exchange Protocol*: To achieve small SNEP messages, we assume that the communicating parties A and B know each other's counter values CA and CB and so the counter does not need to be added to each encrypted message. In practice, however, messages might get lost and the shared counter state can become inconsistent. We now present protocols to synchronize the counter state. To bootstrap the counter values initially, we use the following protocol:

A → B: CA,

B → A: CB, MACK'BACA ‖ CB,

A → B: MACK'AB, CA‖ CB.

.

If party A realizes that the counter CB of party B is not synchronized any more, A can request the current

counter of B using a nonce NA to ensure strong freshness of the reply:

A → B: NA,

B → A: CB, MAC (K'BA, NA‖ CB).

5.3 µTESLA: Authenticated broadcast: Previous proposals for authenticated broadcast are impractical for sensor networks. First, most proposals rely on asymmetric digital signatures for authentication, which are impractical for multiple reasons.

TESLA authenticates the initial packet with a digital signature. Clearly, digital signatures are too expensive to compute on our sensor nodes, since even fitting the code into the memory is a major challenge. For the same reason as we mention above, one-time signatures are a challenge to use on our nodes. Standard TESLA has an overhead of approximately 24 bytes per packet. For networks connecting workstations this is usually not significant. Sensor nodes, however, send very small messages that are around 30 bytes long. It is simply impractical to disclose the TESLA key for the previous intervals with every packet: with 64 bit keys and MACs, the TESLA-related part of the packet would constitute over 50% of the packet. Finally, the one-way key chain doesn'tit into the memory ofour sensor node. So, pureTESLAis notpracticalfora node to broadcast authenticated data. We design µTESLA to solve the following inadequacies of TESLA in sensor networks:

• TESLA authenticates the initial packet with a digital sig- nature, which is too expensive for our sensor nodes. µTESLA uses only symmetric mechanisms.

• Disclosing a key in each packet requires too much energy for sending and receiving. µTESLA discloses the key once per epoch.

• It is expensive to store a one-way key chain in a sensor node. µTESLA restricts the number of authenticated senders

5.4 µTESLA Overview: We give a brief overview of µTESLA, followed by a detailed description.

Authenticated broadcast requires an asymmetric mechanism; otherwise any compromised receiver could forge messages from the sender. Unfortunately, asymmetric cryptographic mechanisms have high computation, communication, and storage overhead, making their usage on resource- constrained devices impractical. µTESLA overcomes this problem by introducing asymmetry through a delayed disclosure of symmetric keys, which results in an efficientbroadcast authentication scheme.

*Example:*µTESLA one-way key chain derivation, the time intervals, and some sample packets that the sender broadcasts. Each key of the key chain corresponds to a time interval and all packets sent within one time interval are authenticated with the same key. In this example, the sender discloses keys two time intervals after it uses them to compute MACs. We assume that the receiver node is loosely time synchronized and knows K0 (a commitment to the key chain). Packets P1 and P2 sent in interval 1 contain a MAC with key K1.

*5.5 µTESLA detailed description:*µTESLA has multiple phases: sender setup, sending authenticated packets, bootstrapping new receivers, and authenticating packets. We first explain how µTESLA allows the base station to broadcast authenticated information to the nodes and we then explain how TESLA allows nodes to broadcast authenticated messages.

M → S: NM

S → M: TS | Ki | Ti | Tint | δ

MAC (KMS, NM |TS | Ki |Ti |Tint | δ).

• The node broadcasts the data through the base station. It uses SNEP to send the data in an authenticated way to the base station, which subsequently broadcasts it.

• The node broadcasts the data. However, the base station keeps the one-way key chain and sends keys to the broad- casting node as needed. To conserve energy for the broad- casting node, the base station can also broadcast the disclosed keys, and/or perform the initial bootstrapping procedure for new receivers.

6. Implementation

Because of stringent resource constraints on the sensor nodes, implementation of the cryptographic primitives is a major challenge. We can sacrifice some security to achieve feasibility and efficiency, but we still need a core level of strong cryptography. Below we discuss how we provide strong cryptography despite restricted resources.

Memory size is a constraint: our sensor nodes have 8 Kbytes of read-only program memory, and 512 bytes of RAM. The program memory is used for TinyOS, our security infrastructure, and the actual sensor net application. To save program memory we implement all cryptographic primitives from one single block cipher

Block cipher: We evaluated several algorithms for use as a block cipher. An initial choice was the AES algorithm Rijn- deal; however, after further inspection, we sought alter- natives with smaller code size and higher speed. The base- line version of Rijndael uses over 800 bytes of lookup tables which is too large for our memory-deprived nodes. An op- timized version of that algorithm (about a 100 times faster) uses over 10 Kbytes of lookup tables. Similarly, we rejected the DES block cipher which requires a 512-entry Box table and a 256-entry table for various permutations. A small encryption algorithm such as TEA is a possibility, but is has not yet been subject to cryptanalytic securutiy.We use RC5 because of its small code size and high efficiency. RC5 does notrely on multiplication and does not require large tables. However, RC5 does use 32-bit data-dependent rotates, which are expensive on our Atmel processor (it only supports an 8-bit single bit rotate operation). Even though the RC5 algorithm can be expressed succinctly, the common RC5 libraries are too large to fit on our

platform. With a judicious selection of functionality, we use a subset of RC5 from Open SSL, and after further tuning of the code we achieve an additional 40% reduction in code size.

*Encryption functio*n: To save code space, we use the same function for both encryption and decryption. The counter (CTR) mode of block ciphers (figure 1) has this property. CTR mode is a stream cipher. Therefore, the size of the cipher text is exactly the size of the plaintext and not a multiple of the block size. This property is particularly desirable in our environment. Message sending and receiving consume a lot of energy.

Also, longer messages have a higher probability of data corruption. Therefore, block cipher message expansion is undesirable. CTR mode requires a counter for proper operation.: the same plaintext sent at different times is encrypted into different cipher text since the encryption pads are generated from different counters. To an adversary who does not know the key, these messages will appear as two unrelated random strings. Since the sender and the receiver share the counter, we do not need to include it in the message. If the two nodes lose the synchronization of the counter, they can simply transmit the counter explicitly to resynchronize using SNEP with strong freshness.

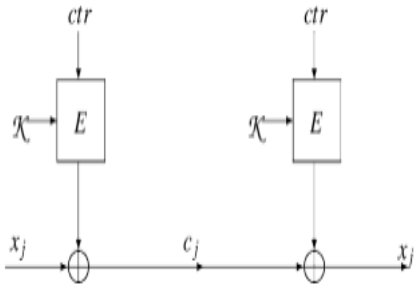

Fig 1:Counter mode encryption and decryption. The encryption function is applied to a monotonically increasing counter to generate a one time pad. This pad is then XORed with the plaintext. The decryption operation is identical.

Key setup: Recall that our key setup depends on a secret master key, initially shared by the base station and the node. We call that shared key XAS for node A and base station S. All other keys are bootstrapped from the initial master secret key.

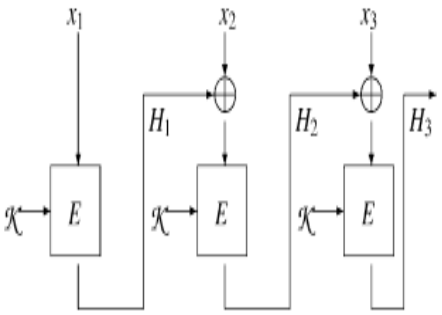

Fig 2 : CBC MAC. The output of the last stage serves as the authentication code

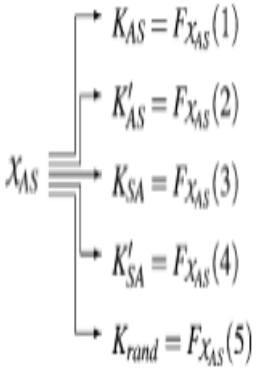

Fig 3: Deriving internal keys from the master secret key

Memory size is a constraint: our sensor nodes have 8 Kbytes of read-only program memory, and 512 bytes of RAM. The program memory is used for TinyOS, our security infrastructure, and the actual sensor net application. Pseudo-random function (PRF) F to derive the keys, which we implement as FK(x) = MAC (K, x). Again, this allows for more code reuse. Because of cryptographic properties of the MAC, it must also be a good pseudo-random function. All keys derived in this manner are computationally independent. Even if the attacker could break one of the keys, the knowledge of that key would not help it find the master secret or any other key. Additionally, if we detect that a key has been compromised, both parties can derive a new key without transmitting any confidentiality information.

7. Evaluation

We evaluate the implementation of our protocols by code size, RAM size, and processor and communication overhead.

7.1 *Code Size*: Table 1 shows the code size of three implementations of crypto routines in TinyOS. The smallest version of the crypto routines occupies about 20% of the available code space. The difference between the fastest and the smallest implementation stems from two different implementations of the variable rotate function. The µTESLA protocol uses another 574 bytes. Together, the crypto library and the protocol implementation consume about 2 Kbytes of program memory, which is acceptable in most applications.

Version	Total size	MAC	Encrypt	Key setup
Smallest	1580	580	402	598
Fastest	1844	728	518	598
Original	2674	1210	802	686

Table 1: Code size breakdown (in bytes) for the security modules.

Operation	Time in ms Fast implementation	Time in ms Small implementation
Encrypt (16 bytes)	1.10	1.69
MAC (16 bytes)	1.28	1.63
Key setup	3.92	3.92

Table 2: Performance of security primitives in Tiny OS.

7.2 Performances: The performance of the cryptographic primitives is adequate for the bandwidth supported by the current generation of network sensors. Key setup is relatively expensive (4 ms). In contrast, the fast version of the code uses less than 2.5 ms to encrypt a 16 byte message and to compute the MAC (the smaller but slower version takes less than 3.5ms). Let us compare these time figures against the speed of our network. Our radio operates at 10 kbps at the physical layer. If we assume that we communicate at this rate, we can perform a key setup, an encryption, and a MAC for every message we send out.

In our implementation, µTESLA discloses the key after two intervals ($\delta = 2$). The stringent buffering requirements also dictate that we cannot drop more than one key disclosure beacon. We require a maximum of two key setup operations and two CTR encryptions to check the validity of a disclosed TESLA key. Additionally, we perform up to two key setup operations, two CTR encryptions, and up to four MAC operation to check the integrity of a TESLA message.7 that gives an upper bound of 17.8 ms for checking the buffered messages. This amount of work is easily performed on our processor. In fact, the limiting factor on the bandwidth of authenticated broadcast traffic is the amount of buffering we can dedicate on individual sensor nodes. Table 4 shows the memory size required by the security modules. We configure the µTESLA protocol with four messages: the disclosure interval dictates a buffer space of three messages just for key disclosure, and we need an additional buffer to use this primitive in a more flexible way

7.3 Energy Costs: We examine the energy costs of security mechanisms. Most energy costs will come from extra trans- missions required by the protocols.

Module	RAM size (bytes)
RC5	80
TESLA	120
Encrypt/MAC	20

Table 3: RAM requirements of the security modules.

71%	Data transmission
20%	MAC transmission
7%	Nonce transmission (for freshness)
2%	MAC and encryption computation

Table 5: Energy costs of adding security protocols to the sensor network. Most of the overhead arises from the transmission of extra data rather than from any computational costs.

8. Applications

In this section we demonstrate how we can build secure proto- cols out of the SPINS secure building blocks. First, we build an authenticated routing application, and second, a two party key agreement protocol.

8.1 *Authenticated Routing:* IN this section we demonstrate how we can build secure proto- cols out of the SPINS secure building blocks. First, we build an authenticated routing application, and second, a two party key agreement protocol.

Using the µTESLA protocol, we developed a lightweight, authenticated ad hoc routing protocol that builds an authenticated routing topology. Ad hoc routing has been an active area of research [11, 20, 25, 26, 38, 40, 41]. Marti et al. discuss a mechanism to protect an ad hoc network against misbehaving nodes that fail to forward packets correctly. They describe two mechanisms: a watchdog to detect misbehaving neighbouring nodes, and a path ratter to keep state about the goodness of other nodes. They propose running these mechanisms on each node. However, we are not aware of a routing protocol that uses authenticated routing messages. It is possible for a malicious user to take over the network by injecting erroneous, replaying old, or advertise incorrect routing information. The authenticated routing scheme we developed mitigates these problems.

The routing scheme within our prototype network assumes bidirectional communication channels, i.e. if node A hears node B, then node B hears node A. The route discovery de-pends on periodic broadcast of beacons. Every node, upon reception of a beacon packet, checks whether it has already received a beacon (which is a normal packet with a globally unique sender ID and current time at base station, protected by a MAC to ensure integrity and that the data is authentic) in the current epoch.8 If a node hears the beacon within the epoch, it does not take any further action. Otherwise, the node accepts the sender of the beacon as its parent to route towards the base station. Additionally, the node would repeat the become with the sender ID changed to itself. This route discovery resembles a distributed, breadth first search algorithm, and produces a routing topology.

8.2 Node-To-Node Key Agreement: A convenient technology for bootstrapping secure connections is to use public key cryptography protocols for symmetric key setup. Unfortunately, our resource constrained sensor nodes prevent us from using computationally expensive public key cryptography. We need to construct our proto-cols solely from symmetric key algorithms. We design a sym- metric protocol that uses the base station as a trusted agent for key setup.

Assume that the node A wants to establish a shared secret session key SKAB with nodeB. Since Aand B do not share any secrets, they need to use a trusted third party S, which is thebase station in our case. In our trust setup, both A and B share a master secret key with the base station, XAS and XBS, respectively. The following protocol achieves secure key agreement as well as strong key freshness:

$$A \rightarrow B: NA, A,$$

$$B \rightarrow S: NA, NB, A, B, MACK'BS, NA|NB|A|B,$$

$$S \rightarrow A: \{SKAB\} KSA, MACK'SA, NA|B| \{SKAB\} KSA,$$

$$S \rightarrow B: \{SKAB\} KSB, MACK'SB,NA|B|\{SKAB\}KSB.$$

The protocol uses our SNEP protocol with strong freshness. The nonce's NA and NB ensure strong key freshness to both A and B. The SNEP protocol ensures confidentiality (throughencryptionwith the keys KAS and KBS)of the established session key SKAB, as well as message authentication (through the MAC using keys KAS and KBS), so we are sure that the key was really generated by the base station. Note that the MAC in the second protocol message helps defend the base station from denial-of-service attacks, and the base station only sends two messages to A and B if it received a legitimate request from one of the nodes.

9. Related work

Tatebayashi et al. consider key distribution for resource-starved devices in a mobile environment. Park et al. point out weaknesses and improvements. Beller and Yacobi further develop key agreement and authentication protocols. Boyd and Mathuria survey the previous work on key distribution and authentication for resource-starved devices in mobile environments .The majority of these approaches rely on asymmetric cryptography. Bergstrom et al. consider the problem of secure remote control of resource-starved de- vices in a home .

Fox and Gribble present a security protocol providing se-cure access to application level proxy services. Their protocol is designed to interact with a proxy to Kerberos and to facilitate porting services relying on Kerberos to wireless devices.

Zhou and Haas propose to secure ad hoc networks using asymmetric cryptography. Recently, Basagni et al. proposed to use a network-wide symmetric key to secure an ad hoc routing protocol. While this approach is efficient, it does not resist compromise of a single node.

Carman et al. analyze a wide variety of approaches for key agreement and key distribution in sensor networks. They analyze the overhead of these protocols on a variety of hardware platforms. Marti et al. discuss a mechanism to protect an ad hoc net- work against misbehaving nodes that fail to forward packets correctly.

10. Conclusion

We designed and built a security subsystem for an extremely limited sensor network platform. We have identified and implemented useful security protocols for sensor networks: authenticated and confidentiality communication, and authenticated broadcast. We have implemented applications including an authenticated routing scheme and a secure node-to-node key agreement protocol.

Most of our design is universal and applicable to other net- works of low-end devices. Our primitives only depend on fast symmetric cryptography, and apply to a wide variety of device configurations. On our limited platform energy spent for security is negligible compared with to energy spent on sending or receiving messages. It is possible to encrypt and authenticate all sensor readings.

The communication costs are also small. Data authentication, freshness, and confidentiality properties use up a net 6 bytes out of 30 byte packets. So, it is feasible to guarantee these properties on a per packet basis. It is difficult to improve on this scheme, as transmitting a MAC is fundamental to guaranteeing data authentication.

Certain elements of the design were influenced by the available experimental platform. If we had a more powerful platform, we could have used block ciphers other than RC5. The emphasis on code reuse is another property forced by our platform. A more powerful device would allow more modes of authentication. In particular, memory restrictions on buffering limit the effective bandwidth of authenticated broadcast.

Despite the shortcomings of our target platform, we built a system that is secure and works.

With our techniques, we believe security systems can become an integral part of practical sensor networks.

References

1) Atmel, Secure Microcontrollers for Smartcards, http://www. atmel.com/atmel/acrobat/1065s.pdf
2) S. Basagni, K. Herrin, E. Rosti and D. Bruschi, Secure Pebblenets, in: ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001) (2001) pp. 156–163.
3) M. Bellare, A. Desai, E. Jokipii and P. Rogaway, A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation, in: Symposium on Foundations of Computer Science (FOCS) (1997).
4) M. Beller and Y. Yacobi, Fully fledged two-way public key authentication and key agreement for low-cost terminals, Electronics Letters 29(11) (1993) 999–1001.
5) S. Bellovin and M. Merrit, Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise, in: ACM Conference on Computer and Communications Security CCS-1 (1993) pp. 244–250.

