Wireless Networks 8, 521–534, 2002

SPINS: Security Protocols for Sensor Networks

ADRIAN PERRIG, ROBERT SZEWCZYK, J.D. TYGAR, VICTOR WEN and DAVID E. CULLER
Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 387 Soda Hall, Berkeley, CA 94720, USA

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 data authentication, 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:

- *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.
- *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.
- *Medical monitoring*: we envision a future where individuals with some types of medical conditions receive constant monitoring through sensors that monitor health conditions. For some types of medical conditions, remote sensors may apply remedies (such as instant release of emergency medication to the bloodstream).
- *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.
- *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, and vehicles in 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:

- Exploring the challenges for security in sensor networks.
- Designing and developing μTESLA (the "micro" version of TESLA), providing authenticated streaming broadcast.
- Designing and developing SNEP (*Secure Network Encryption Protocol*) providing data confidentiality, two-party data authentication, and data freshness, with low overhead.
- 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 SmartDust program [45], one of the components of CITRIS. We have deployed these in one of

Table 1
Characteristics of prototype SmartDust nodes.

CPU	8-bit, 4 MHz
Storage	8 Kbytes instruction flash
	512 bytes RAM
	512 bytes EEPROM
Communication	916 MHz radio
Bandwidth	10 Kbps
Operating system	TinyOS
OS code space	3500 bytes
Available code space	4500 bytes

our EECS buildings, Cory Hall. We are currently using these for a very simple application – heating and air-conditioning control in the building. However, the same mechanisms that we describe in this paper can be modified to support sensor that handle emergency system such as fire, earthquake, and hazardous material response.

By design, these sensors are inexpensive, low-power devices. As a result, they have limited computational and communication resources. The sensors form a self-organizing wireless network and form a multihop routing topology. Typical applications may periodically transmit sensor readings for processing.

Our current prototype consists of *nodes*, small battery powered devices, that communicate with a more powerful *base station*, which in turn is connected to an outside network. Table 1 summarizes the performance characteristics of these devices. At 4 MHz, they are slow and underpowered (the CPU has good support for bit and byte level I/O operations, but lacks support for many arithmetic and some logic operations). They are only 8-bit processors (note that according to [53], 80% of all microprocessors shipped in 2000 were 4 bit or 8 bit devices). Communication is slow at 10 Kbps.

The operating system is particularly interesting for these devices. We use TinyOS [23]. 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 performed by these 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 (battery replacement is designed to reinitialize devices and zero out keys).[1]

1.2. Is security on sensors possible?

These constraints make it impractical to use most current secure algorithms, since they were designed for powerful processors. For example, the working memory of a sensor node is not sufficient to even hold the variables for asymmetric cryptographic algorithms (e.g., RSA [48] with 1024 bits), let alone perform operations with them.

A particular challenge is broadcasting authenticated data to the entire sensor network. Current proposals for authenticated broadcast are impractical for sensor networks. Most proposals rely on asymmetric digital signatures for the authentication, which are impractical for multiple reasons (e.g., long signatures with high communication overhead of 50–1000 bytes per packet, very high overhead to create and verify the signature). Furthermore, previously proposed purely symmetric solutions for broadcast authentication are impractical: Gennaro and Rohatgi's initial work required over 1 Kbyte of authentication information per packet [17], and Rohatgi's improved *k*-time signature scheme requires over 300 bytes per packet [49]. Some of the authors of this article have also proposed the authenticated streaming broadcast *TESLA* protocol [43]. TESLA works well on regular desktop workstations, but uses too much communication and memory on our resource-starved sensor nodes. This article extends and adapts TESLA to make it practical for broadcast authentication for sensor networks. We call our new protocol μTESLA.

We have implemented all of these primitives. Our measurements show that adding security to a highly resource-constrained sensor network is feasible.

Given the severe hardware and energy constraints, we must be careful in the choice of cryptographic primitives and the security protocols in the sensor networks.

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 network, 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.

A typical SmartDust sensor network forms around one or more *base stations*, which interface the sensor network to the outside network. The sensor nodes establish a routing forest, with a base station at the root of every tree. Periodic transmission of beacons allows nodes to create a routing topology. Each node can forward a message towards a base station, recognize packets addressed to it, and handle message broadcasts. The base station accesses individual nodes using source routing. We assume that the base station has capabilities similar to the network nodes, except that it has sufficient battery power to surpass the lifetime of all sensor nodes, sufficient memory to store cryptographic keys, and means for communicating with outside networks.

[1] Base stations differ from nodes in having longer-lived energy supplies and additional communications connections to outside networks.

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. Trust requirements

Generally, the sensor networks may be deployed in untrusted locations. While it may be possible to guarantee the integrity of the each node through dedicated secure microcontrollers (e.g., [1] or [13]), we feel that such an architecture is too restrictive and does not generalize to the majority of sensor networks. Instead, we assume that individual sensors are untrusted. Our goal is to design the SPINS key setup so a compromise of a node does not spread to other nodes.

Basic wireless communication is not secure. Because it is broadcast, any adversary can eavesdrop on traffic, inject new messages, and replay old messages. Hence, our protocols do not place any trust assumptions on the communication infrastructure, except that messages are delivered to the destination with non-zero probability.

Since the base station is the gateway for the nodes to communicate with the outside world, compromising the base station can render the entire sensor network useless. Thus the base stations are a necessary part of our trusted computing base. Our trust setup reflects this and so all sensor nodes intimately trust the base station: at creation time, each node gets a *master secret key* X which it shares with the base station. All other keys are derived from this key, as we show in section 6.

Finally, each node trusts itself. This assumption seems necessary to make any forward progress. In particular, we trust the local clock to be accurate, i.e. to have small drift. This is necessary for the authenticated broadcast protocol we describe in section 5.

2.3. Design guidelines

With the limited computation resources available on our platform, we cannot afford to 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 primitives (i.e. encryption, message authentication code (MAC), hash, random number generator) out of a single block cipher for code reuse. To reduce communication overhead we exploit common state between the communicating parties.

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 neighboring 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 network 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.

In the two-party communication case, data authentication can be achieved through a purely symmetric mechanism: The sender and the receiver share a secret key to compute a message authentication code (MAC) of all communicated data. When a message with a correct MAC arrives, the receiver knows that it must have been sent by the sender.

This style of authentication cannot be applied to a broadcast setting, without placing much stronger trust assumptions on the network nodes. If one sender wants to send authentic data to mutually untrusted receivers, using a symmetric MAC is insecure: any one of the receivers knows the MAC key, and hence, could impersonate the sender and forge messages to other receivers. Hence, we need an asymmetric mechanism to achieve authenticated broadcast. One of our contributions is to construct authenticated broadcast from symmetric primitives only, and introduce asymmetry with delayed key disclosure and one-way function key chains.

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