SPINS: Security Protocols for Sensor Networks*

Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, J. D. Tygar

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

{perrig, szewczyk, vwen, culler, tygar}@cs.berkeley.edu

ABSTRACT

As sensor networks edge closer towards wide-spread deployment, security issues become a central concern. So far, the main research focus has been on making sensor networks feasible and useful, and less emphasis was placed on security.

We design a suite of security building blocks that are optimized for resource-constrained environments and wireless communication. SPINS has two secure building blocks: SNEP and μTESLA. SNEP provides the following important baseline security primitives: Data confidentiality, two-party data authentication, and data freshness. A particularly hard problem is to provide efficient broadcast authentication, which is an important mechanism for sensor networks. μTESLA is a new protocol which provides authenticated broadcast for severely resource-constrained environments. We implemented the above protocols, and show that they are practical even on minimalistic 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.

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, storage, bandwidth, and energy.

Despite the challenges, security is important for these devices. As we describe below, we are deploying prototype wireless network sensors at UC Berkeley. These sensors measure environmental parameters and we are experimenting with having them control air conditioning and lighting systems. Serious privacy questions arise if third parties can read or tamper with sensor data. In the future, we envision wireless sensor networks being used for emergency and life-critical systems – and here the questions of security are foremost.

This paper presents a set of *Security Protocols for Sensor Networks*, SPINS. The chief contributions of this paper are:

Our main contributions include:

- Exploring the challenges for security in sensor networks.
- Designing and developing μTESLA (the "micro" version of the *Timed, Efficient, Streaming, Loss-tolerant Authentication Protocol*), 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 SPINS building blocks

1.1 Sensor Hardware

At UC Berkeley, we are building prototype networks of small sensor devices under the SmartDust program [32]. We've deployed these in one of our EECS buildings, Cory Hall (see Figure 1). 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. As described below, the sensors form a self-organizing network (see Figure 1). Table 1 summarizes the performance characteristics of these devices. At 4MHz, 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 logic operations). They are only 8-bit processors (note that according to [40], 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 [16]. This small, event-driven operating system consumes ab operating system consumes almost half of 8KB of instruction flash memory, leaving just 4500 bytes for security and the application.

*We gratefully acknowledge funding support for this research. This research was sponsored in part the United States Postal Service (contract USPS 102592-01-Z-0236), by the United States Defense Advanced Research Projects Agency (contracts DABT63-98-C-0038, "Ninja", N66001-99-2-8913, "Endeavour", and DABT63-96-C-0056, "IRAM"), by the United States National Science Foundation (grants FD99-79852 and R1 EIA-9802069) and from gifts and grants from the California MICRO program, Intel Corporation, IBM, Sun Microsystems, and Philips Electronics. DARPA Contract N66001-99-2-8913 is under the supervision of the Space and Naval Warfare Systems Center, San Diego. This paper represents the opinions of the authors and do not necessarily represent the opinions or policies, either expressed or implied, of the United States government, of DARPA, NSF, USPS, or any other of its agencies, or any of the other funding sponsors.

Mobile Computing and Networking 2001 Rome, Italy

CPU	8-bit, 4MHz
Storage	8KB instruction flash
	512 bytes RAM
	512 bytes EEPROM
Communication	916 MHz radio
Bandwidth	10Kilobits per second
Operating System	TinyOS
OS code space	3500 bytes
Available code space	4500 bytes

Table 1: Characteristics of prototype SmartDust Nodes

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. Similarly, since communication over radio will be the most energy-consuming function performed by these devices, we need to minimize communications overhead. The limited energy supplies creates tensions for security: on the one hand, security needs to limit its consumption of processor power; on the other hand, limited power supply limits key lifetime (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 the majority of the current secure algorithms, which were designed for powerful workstations. For example, the working memory of a sensor node is insufficient to even hold the variables (of sufficient length to ensure security) that are required in asymmetric cryptographic algorithms (*e.g.* RSA [35], Diffie-Hellman [8]), 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. First, 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).

Broadcast authentication is another problem. Even 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 [11], and Rohatgi's improved k-time signature scheme requires over 300 bytes per packet [36]. Some of the authors have also proposed the authenticated streaming broadcast *TESLA* protocol [31], and TESLA is efficient for the Internet with regular desktop workstations, but does not scale down to our resource-starved sensor nodes. In this paper, we extend and adapt TESLA such that it becomes practical for broadcast authentication for sensor networks. We call our new protocol μTESLA.

We've implemented all of these primitives, Our measurements show that adding security to a highly resource-constrained sensor network is feasible. The paper studies an authenticated routing protocol and a two-party key agreement protocol, and demonstrates that our security building blocks greatly facilitate the implementation of a complete security solution for a sensor network.

A common characteristic of sensor networks is their severely limited energy supply. Ultimately, the available energy determines the amount of computation, sensing, and communication a node can perform in its lifetime. Alternatively, the power harvested from the environment sets a bound on computation and communication per unit of time. In order to minimize the energy usage, a security subsystem should place minimal requirements on the processor, and add minimal information to each message transmitted. On the other hand, the limited lifespan of each node limits the life time of usable keys; we think of the battery replacement process as a rebirth.

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.

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

2. SYSTEM ASSUMPTIONS

Before we outline the security requirements and present our security infrastructure, we need to define our system architecture and the trust setup. The goal of this work is to propose a general security infrastructure that is applicable to a variety of sensor networks. Hence we chose a minimal hardware infrastructure as a basis for our design, such that SPINS can scale up to arbitrary sensor networks.

Sensor Hardware

The sensor nodes used in this design have the computational power and storage capacity comparable to that of the earliest PCs. The CPU is a RISC-like, 8-bit processor with 32 general purpose registers. This processor runs at a speed to 4MHz with the CPI of 2. The instruction set architecture is quite limited: it has a good support for bit- and byte-level I/O operations, but lacks support for many arithmetic and logic operations. The total amount of storage onboard is 8KB of instruction flash, 512 bytes of data RAM and 512 bytes of EEPROM. Every node is equipped with a short-range, 916MHz ISM band radio with 10Kbps of bandwidth. Each node is running a small event-driven operating system called TinyOS [16]. A typical sensor network application establishes a multihop routing topology and periodically transmits unprocessed sensor readings. Such an application uses about 3500 bytes of code space for TinyOS, which leaves at most 4500 bytes for security and the application. As technology improves, we expect that sensor networks have devices with similar capability, but in a smaller form factor.

A common characteristic of sensor networks is their severely limited energy supply. Ultimately, the available energy determines the amount of computation, sensing, and communication a node can perform in its lifetime. Alternatively, the power harvested from the environment sets a bound on computation and communication per unit of time. In order to minimize the energy usage, a security subsystem should place minimal requirements on the processor, and add minimal information to each message transmitted. On the other hand, the limited lifespan of each node limits the life time of usable keys; we think of the battery replacement process as a rebirth.

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.

Communication architecture

Generally, the sensor nodes communicate using RF, so broadcast is the fundamental communication primitive. The baseline protocols account for this property: on one hand it affects the trust assumptions, and on the other it is exploited to minimize the energy usage.

Figure 1 shows the organization of a typical SmartDust sensor network. The network forms around one or more *base stations*,

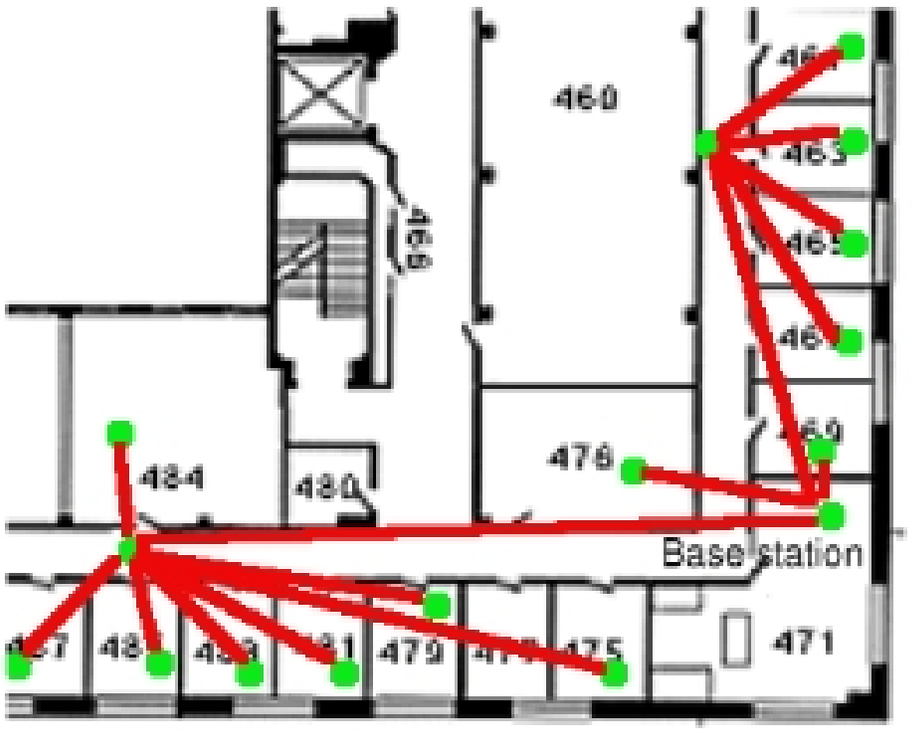

Figure 1: Communication organization within a sensor network. All messages are either destined for the base station or originate at the base station. The routes are discovered so that the number of hops is minimized and the reliability of each connection is maximized.

which interface the sensor network to the computing infrastructure. 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 enough battery power to surpass the lifetime of all sensor nodes, sufficient memory to store cryptographic keys, and means for communicating with outside networks.

In the sensor applications developed so far, there has been limited local exchange and data processing. 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 primarily these communication patterns, though we do show how to adapt our baseline protocols to other communication patterns, i.e. node to node or node broadcast.

Trust Setup

Generally, the sensor networks may be deployed in untrusted locations. While it is conceivable to guarantee the integrity of the each node through dedicated secure microcontrollers (*e.g.* [1] or [7]), 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 such that a compromised sensor only compromises that sensor, and no other sensors of the network.

Wireless communication is fundamentally untrusted. Because of its broadcast nature any adversary can eavesdrop on the traffic, and inject new messages or replay and change old messages. Hence, SPINS does 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 could render the entire sensor network useless. Thus the base stations are a necessary part of our trusted computing base. Our trust setup mimics this and so all sensor nodes intimately trust the base station: at creation time, each node is given a *master key* which is shared with the base station. All other keys are derived from this key.

Finally, each node trusts itself and its sensors. This assumption seems necessary to make any forward progress. In particular, we trust the local clock to be accurate, *i.e.* to have a small drift. This assumption is necessary for the authenticated broadcast protocol described below in Section 5.2.

Design guidelines

With the limited computation resources available on our platform, we cannot afford to use asymmetric cryptography and hence we use purely 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

In this section, we formalize the security properties required by sensor networks, and show how they are directly applicable in a sample network deployed within a typical building.

Data Confidentiality

A sensor network within an apartment should not leak sensor readings to the neighboring networks. In many applications (*e.g.* key distribution) the nodes communicate highly sensitive data. The standard solution to keep sensitive data secret is to encrypt the data with a secret key that only the intended receivers possess, hence achieving confidentiality. Given the observed communication patterns, we use initially set up secure channels between nodes and base stations to bootstrap other secure channels, if necessary.

Data Authentication

Message authentication is of paramount importance for many applications in sensor networks. Within the building sensor network, authentication is necessary for many administrative tasks (*e.g.* network reprogramming or controlling sensor node duty cycle). At the same time, an adversary can easily inject messages, so the receiver needs to make sure that the data used in any decision-making process originates from the coerect source. Informally, *data authentication* allows the 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. Our