SNAP: An Architecture for Secure Medical Sensor Networks

Kriangsiri Malasri and Lan Wang, *Computer Science Dept., University of Memphis*

Abstract—**We identify the security threats facing a sensor network for wireless medical monitoring, and we propose a public-key architecture using elliptic curve cryptography to address these issues. We also present a preliminary protocol for securely establishing pairwise symmetric keys between a sensor and a base station using ECC. To evaluate our protocol, we are working on an implementation of ECC for the Moteiv Tmote Sky platform. Our initial release performs competitively with existing implementations for the Crossbow MICA.**

I. Introduction

DURING the past few years, wireless sensor technology has shown great potential as an enabler of the vision of ubiquitous computing. One promising application of wireless sensor networks (WSNs) is healthcare. In a traditional hospital setting, a patient is attached to a number of medical sensors that convey information on his or her vital signs to a bedside monitoring device. However, because these connections are generally wired, such a setup is cumbersome for the patient and limits mobility, especially for long-term care. Recent advances in sensor technology have enabled the development of small, lightweight medical sensors such as pulse oximeters and electrocardiogram leads [12] that can be worn by the patient while wirelessly transmitting data. This frees the patient from the confinement of traditional wired sensors, allowing him or her to move about the hospital area at leisure and increasing comfort.

Medical sensor networks present several challenges. First, while most previous research assumes stationary sensors, a medical sensor network may contain a large number of mobile sensors due to the mobility of patients. Second, the timeliness and reliability of data delivery is crucial in detecting and diagnosing health problems, but wireless sensor networks generally have limited bandwidth and high loss rate. ***The most fundamental challenge is the security and privacy of sensitive patient data.*** Because the data is transmitted wirelessly, it is easy for an eavesdropper with a properly tuned receiver to intercept the data. Encryption must be done to ensure the *confidentiality* of the data. At the same time, a sensor receiving a query from a base station (and likewise a base station receiving data from a sensor) needs to have some way of verifying the identity of the other party, and of ensuring that the data has not been altered from its source. Hence, mechanisms must exist for data *authenticity* and *integrity*. What makes security uniquely challenging for WSNs is that *the computational, memory and bandwidth costs must be carefully balanced against the limited resources of the individual nodes*.

We have recently started a project called **SNAP (Sensor Network for Assessment of Patients)** to address the above challenges. As our first step, we explore potential solutions to the security concerns based on elliptic curve cryptography (ECC). Although the prevailing sentiment in the research community has been that public-key schemes such as ECC are infeasible for WSNs due to their high computational cost, recent work has shown that performing public-key computations on resource-constrained devices, while relatively expensive, is certainly not impossible. ECC in particular is promising because it offers a similar level of security to RSA for a much smaller key length, thus allowing more efficient operations.

Our current work on SNAP focuses on the following security aspects:

- An in-depth analysis of the specific security threats to a medical sensor network.
- Development of a secure protocol using ECC for key establishment and updates in medical sensor networks, which takes into account the unique requirements of sensor networks in this environment.
- ECC implementation on the Moteiv Tmote Sky platform rather than the Crossbow MICA popular in the literature. We believe that the Tmote Sky offers the potential for superior performance to the MICA.

II. Related Work

The CodeBlue project at Harvard has proposed a sensor network platform for medical care [12]. Although the authors acknowledge the need for security in a medical environment and have considered ECC in [3], their work has yet to adequately address all the security requirements. Their ECC implementation over the binary field $F_{2^{163}}$ (i.e. key length of 163 bits) takes 34.2 s to compute a public-private key pair and another 34.2 s to compute a shared secret via ECDH [3].

Early work on sensor network security focused on applying symmetric cryptography (e.g. [1]), as it was felt that asymmetric cryptography schemes are too computationally expensive to be feasible in resource-constrained sensor nodes. However, symmetric schemes offer less flexibility for key management, and key distribution becomes a formidable challenge.

The use of public-key cryptography (PKC) addresses the issue of key distribution. We can use PKC techniques to securely establish a shared secret between two parties, who then use the secret to encrypt and decrypt transmitted data with a traditional symmetric scheme. Hence the computational burden of PKC is limited to initial key

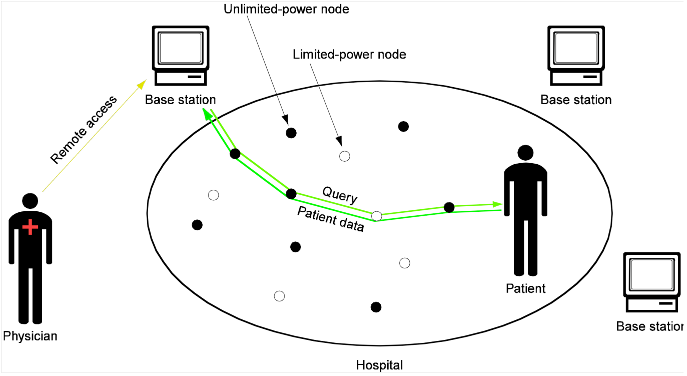

Fig. 1. Architecture of SNAP for wireless medical monitoring.

establishment and key updates. Recent work on PKC in the context of sensor networks has included implementations of both RSA [2] and ECC [3][4][5]. Blaβ and Zitterbart [4] implemented various ECC schemes over the binary field $F_{2^{113}}$ (key length of 113 bits), reporting times of 6.9s and 24.2s for ECDSA signature generation and verification, respectively. Liu and Ning [5] opted to use prime fields rather than binary fields, implementing ECC over various 160-bit prime fields (equivalent to a 1024-bit RSA key) recommended by SECG [10]. They report times of 6.1s and 12.2s for ECDSA signature generation and verification.

Gupta et al [6] went so far as to create a Web server implementing an ECC version of SSL runnable on sensor motes. Their results are by far the most impressive, taking less than 4 seconds to complete an entire SSL handshake. However, this scheme uses ECDH to derive a shared key, which requires both the client and the server to have public keys. Our work does not assume that each sensor has a public key, as explained in the next section.

III. Security Requirements

The particular threats facing a medical sensor network include: (1) eavesdropping on data by an unauthorized third party; (2) modification and injection of data by a third party without the knowledge of the source or destination; (3) replay of previous queries/data; (4) spoofing of a base station to obtain illegitimate access to data; (5) spoofing of a sensor to report forged data; (6) compromise of sensors (or the patients may simply lose their sensors while moving around); (7) compromise of base stations; and (8) compromise of relay nodes. We consider all these threats in our work. However, we realize that in certain cases the most we can do is to limit damage (e.g. compromised relay nodes may simply drop packets).

Our SNAP architecture is shown in Fig. 1. Each patient has one or more wireless sensors attached to his or her body. There are a number of wireless relay nodes throughout the hospital area that can receive and forward data. These relay nodes may be supplied with continuous power (unlimited power) or powered by batteries (limited power). Queries for patient data can be made from a number of base stations, which may be operated directly by medical professionals or connected to remotely. Both queries and the resulting patient data response travel through the network of relay nodes. All the queries and data are encrypted on an *end-to-end* basis. While on the surface our architecture is similar to the one proposed by CodeBlue [12], there are several key differences due to our emphasis on security.

First, the sensors only accept queries from the base stations. This is because in a large hospital, it would be difficult for the sensors to authenticate individual users (doctors, nurses, staff, etc.). Each sensor would have to maintain complete access control information about which users have the authorization to receive its data as well as the users' identity information (e.g. public key or password). The sensors simply do not have the memory resources to maintain all this information. Moreover, it would be infeasible to update the access control information in every sensor. Therefore, we decide to authenticate the users at the base stations and have the base stations issue the queries to the sensors on behalf of the users. **Second, in order to handle spoofing of the base stations, the sensors are equipped with the base stations' public keys. However, we do not assume that each sensor has its own public/private key pair.** Since there may be a large number of sensors compared to base stations, requiring each sensor to have its own public key would force the base stations to maintain a large number of sensor public keys, thus resulting in poor scalability. Furthermore, individual sensors are relatively easy to physically compromise. Doing so when the sensor possesses a permanent private key allows an attacker to easily decrypt pre-captured data of that sensor. **Third, a base station will not accept data from a sensor until the sensor is attached to a patient registered in the system.** To verify that a sensor is attached to a registered patient, each sensor will be integrated with a small fingerprint reader (similar to [13]), so that the sensor can transmit the patient's fingerprint signature to the base station. We assume the base stations are equipped with a list of valid patient fingerprint signatures.

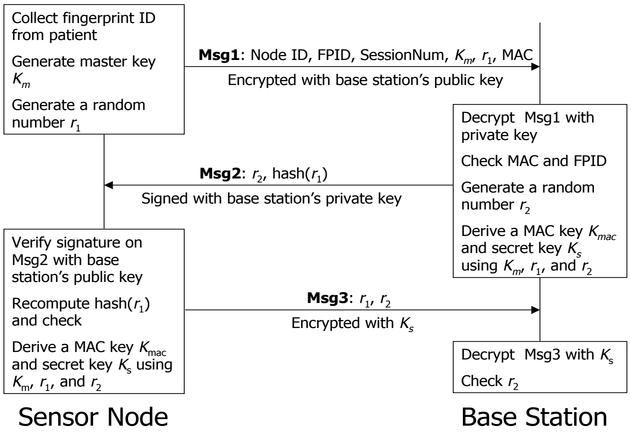
Collect fingerprint ID from patient
Generate master key K_m
Generate a random number r_1
Msg1: Node ID, FPID, SessionNum, K_m, r_1, MAC
Encrypted with base station's public key
Decrypt Msg1 with private key
Check MAC and FPID
Generate a random number r_2
Derive a MAC key K_{mac} and secret key K_s using K_m, r_1, and r_2
Msg2: r_2, hash(r_1)
Signed with base station's private key
Verify signature on Msg2 with base station's public key
Recompute hash(r_1) and check
Derive a MAC key K_{mac} and secret key K_s using K_m, r_1, and r_2
Msg3: r_1, r_2
Encrypted with K_s
Decrypt Msg3 with K_s
Check r_2
Sensor Node
Base Station

Fig. 2. Handshake protocol for secure establishment of keys between a sensor and a base station.

IV. Protocol for Secure Key establishment

We offer the following preliminary protocol for secure key establishment between a patient sensor and a base station (see Fig. 2). When a sensor is attached to a patient, the patient uses a fingerprint sensor to initiate the key generation procedure. The fingerprint uniquely identifies the patient. Once the patient's fingerprint ID (FPID) is obtained, the sensor generates a random number r_1 and a master key K_m.

Message 1 (KeyGenStart). The sensor sends a message to the base station consisting of the following information: a node ID (NID) uniquely identifying the sensor, the FPID, a session number to prevent replay attacks, K_m, and r_1. In addition, a message authentication code (MAC) is computed over the contents of the message and appended. This entire message is encrypted with the base station's public key P_B. Upon receipt of Message 1, the base station decrypts it using its private key n_B and recomputes the MAC. If the MAC check succeeds and the FPID in the message matches with a registered patient's FPID, the base station generates a random number r_2. It uses r_1 and r_2, in conjunction with K_m, to derive two new keys K_s and K_{mac}. Note that the FPID check mechanism assumes that the space of possible FPIDs is large enough that it is infeasible for an attacker to spoof a patient by a brute-force search of FPIDs.

Message 2 (KeyGenAck). The base station replies to the sensor with a message containing r_2 and a one-way hash of r_1. This message is unencrypted, but it is signed using n_B. Upon receipt of Message 2, the sensor checks that the hash of r_1 is correct and verifies the signature using P_B. If both checks succeed, the sensor uses K_m, r_1, and r_2 to derive K_s and K_{mac}.

Message 3 (KeyGenVerify). The sensor constructs a reply to send to the base station containing r_1 and r_2. This message is encrypted with K_s. Upon receipt of Message 3, the base station verifies that r_2 is correct. If so, data transfer may commence using K_s to encrypt/decrypt messages and, if desired, K_{mac} to compute a keyed MAC for each message.

Updating the shared secret keys K_s and K_{mac} may take place by exchanging new values of r_1 and r_2. These key update messages can be encrypted with K_s.

V. Implementation

In order for us to evaluate our protocol, we need an implementation of ECC. This is a work nearing completion; it is based largely on TinyECC [5]. TinyECC incorporates several optimizations, including Jacobian coordinates for minimizing field inversions [11], the sliding window method for reducing point additions [8], a fast modular reduction algorithm for pseudo-Mersenne field primes [7], and inline assembly code for several critical field operations on multiprecision numbers. The authors of TinyECC report running times of 6.1 s for ECDSA signature generation and 12.2 s for signature verification using 160-bit keys, on the 8-bit Atmel ATmega128 CPU of Crossbow's MICAz platform.

We are designing our ECC implementation on the Moteiv Tmote Sky, which uses the 16-bit Texas Instruments MSP430. The Tmote Sky offers more RAM (10 KB vs. 4 KB) and a much larger flash memory for data storage (1 MB vs. 512 KB) compared to the MICA, but at the cost of smaller read-only program memory (48 KB vs. 128 KB).

We have modified TinyECC to run on Tmote Sky by replacing the ATmega128 assembly code with MSP430 assembly, making use of the MSP430's hardware multiplier, incorporating a fast modular inversion algorithm involving only bit shifts and additions [14], and making some changes to the timers used. Our initial version of the code carries out a 160-bit scalar point multiplication in 5.3 s, which is comparable to TinyECC's result.

We plan to incorporate several additional optimizations in our ECC implementation. One optimization not considered by TinyECC is the use of alternate representations of the multiplier in a scalar point multiplication, known as non-adjacent forms or more generally *w*-NAFs; such techniques promise a 10-20% performance improvement [9]. Additionally, TinyECC does not use Shamir's method to compute the sum of two scalar multiplications in the signature verification step of ECDSA. Implementing this simple algorithm will halve the number of point doublings required to compute the sum. One more optimization that may be made is to increase the size of the window used in the sliding window method of [8], to take advantage of the 1 MB of flash memory on the Tmote Sky.

VI. Summary

We have proposed a public-key based architecture and security protocol to achieve confidentiality, authenticity, and integrity for a medical sensor network. Our current focus is to evaluate the protocol by implementing ECC with the Moteiv Tmote Sky platform in mind. We hope that the 16-bit processor on the Tmote Sky, along with the larger RAM and flash memory available compared to the Crossbow MICA, will allow us to achieve an efficient implementation.

References

[1] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: Security Protocols for Sensor Networks," ACM MOBICOM, 2001.

[2] R. Watro, D. Kong, S.-F. Cuti, C. Gardiner, C. Lynn, and P. Kruus, "TinyPK: Securing Sensor Networks with Public Key Technology," ACM SASN, Oct. 2004.

[3] D. J. Malan, M. Welsh, and M. D. Smith, "A Public-Key Infrastructure for TinyOS Based on Elliptic Curve Cryptography," IEEE Int. Conf. on Sensor and Ad Hoc Communications and Networks, Oct. 2004.

[4] E.-O. Blaβ and M. Zitterbart, "Towards Acceptable Public-Key Encryption in Sensor Networks," Int. Workshop on Ubiquitous Computing, ACM SIGMIS, May 2005.

[5] A. Liu and P. Ning, "TinyECC: Elliptic Curve Cryptography for Sensor Networks," v0.1, http://discovery.csc.ncsu.edu/software/TinyECC/, Sept. 2005.

[6] V. Gupta, M. Millard, S. Fung, Y. Zhu, N. Gura, H. Eberle, and S. C. Shantz, "Sizzle: A Standards-Based End-to-End Security Architecture for the Embedded Internet," IEEE Int. Conf. on Pervasive Computing and Communication, Mar. 2005.

[7] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs," Workshop on Cryptographic Hardware and Embedded Systems (CHES), Aug. 2004.

[8] Ç. K. Koç, "High-Speed RSA Implementation," v2.0, RSA Laboratories, Nov. 1994.

[9] V. Dimitrov, L. Imbert, and P. K. Mishra, "Efficient and Secure Elliptic Curve Multiplication using Double-Base Chains," ASIACRYPT, 2005.

[10] "SEC 2: Recommended Elliptic Curve Domain Parameters," v1.0, Standards for Efficient Cryptography Group, Sept. 2000.

[11] H. Cohen, A. Miyaji, and T. Ono, "Efficient Elliptic Curve Exponentiation using Mixed Coordinates," ASIACRYPT, 1998.

[12] V. Shnayder, B.-R. Chen, K. Lorincz, T. R. F. Fulford-Jones, and M. Welsh, "Sensor Networks for Medical Care," Technical Report TR-08-05, Division of Engineering and Applied Sciences, Harvard Univ., 2005.

[13] Fujitsu MBF200 Solid State Fingerprint Sensor, http://www.fujitsu.com/emea/services/microelectronics/sensors/.

[14] S. Chang Shantz, "From Euclid's GCD to Montgomery Multiplication to the Great Divide," Technical Report TR-2001-95, Sun Microsystems, June 2001.