9-19-2011

Hide-n-Sense: Privacy-aware secure mHealth sensing

Shrirang Mare
Dartmouth College

Jacob Sorber
Dartmouth College

Minho Shin
Myongji University

Cory Cornelius
Dartmouth College

David Kotz
Dartmouth College

Hide-n-Sense: Privacy-aware secure mHealth sensing

Dartmouth Computer Science Technical Report TR2011-702

Shrirang Mare[1], Jacob Sorber[1], Minho Shin[2], Cory Cornelius[1], and David Kotz[1]

[1]*Institute for Security, Technology, and Society, Dartmouth College, USA*
[2]*Department of Computer Engineering, Myongji University, South Korea*

September 19, 2011

Abstract

As healthcare in many countries faces an aging population and rising costs, mobile sensing technologies promise a new opportunity. Using mobile health (mHealth) sensing, which uses medical sensors to collect data about the patients, and mobile phones to act as a gateway between sensors and electronic health record systems, caregivers can continuously monitor the patients and deliver better care. Furthermore, individuals can become better engaged in monitoring and managing their own health. Although some work on mHealth sensing has addressed security, achieving strong privacy for low-power sensors remains a challenge.

We make three contributions. First, we propose an mHealth sensing protocol that provides strong security and privacy properties with low energy overhead, suitable for low-power sensors. The protocol uses three novel techniques: adaptive security, to dynamically modify transmission overhead; MAC striping, to make forgery difficult even for small-sized MACs; and an asymmetric resource requirement. Second, we demonstrate a prototype on a Chronos wrist device, and evaluate it experimentally. Third, we provide a security, privacy, and energy analysis of our system.

1 Introduction

As healthcare systems struggle worldwide to cope with increasing demand (due to an aging population) and economic pressures (due to rising costs), many countries are seeking new models of healthcare. Recent improvements in mobile computing and developments in miniature medical sensors have enabled mobile health (mHealth) sensing, promising new opportunities to simultaneously reduce cost and improve the quality of healthcare [29]. In an mHealth sensing system, patients wear or carry one or more sensing devices, and their mobile phone acts as a gateway between the sensors and a repository that makes the data accessible to patients or their caregivers.

An mHealth sensing system can deliver continuous health monitoring to patients throughout their daily activities, providing timely reports about the patients' medical condition, close monitoring by healthcare providers, and reduced re-hospitalizations [8]. Clinical applications include long-term care for patients with chronic diseases [1] or risk management for people in rehabilitation [35]. There are also many non-clinical applications for mHealth sensing, including elder care [2], lifestyle coaching for people seeking to change unhealthy behavior [7], and fitness monitoring for athletes [11]. (Although the monitored individual in these applications may be a resident of an assisted-living facility, a family member, an athlete, or simply yourself, for the purposes of this paper we always refer to the subject of sensing as the "patient.") The security and privacy challenges are largely the same, at the point of sensing; in this paper we focus on the *mobile* edge of the mHealth ecosystem and thus the solutions we propose in this paper apply equally well to all above settings.

Whatever the setting, mHealth sensing collects personal health-related information, inside and outside the hospital setting, in various activities of daily living. Such data is inherently sensitive, and its unintended disclosure would violate the patient's privacy and possibly cause social or economic harm to the patient. Furthermore, since this data may be used to diagnose a patient's condition or adjust a patient's treatment, any data corruption caused by an adversary

This technical report is an expanded version of the paper that is to appear in the Proceedings of the ACM Workshop on Privacy in the Electronic Society (WPES), October 2011.

can have an adverse effect on the patient's health. Thus, an mHealth sensing system should have mechanisms to preserve patient privacy and maintain data integrity.

To achieve the necessary security and privacy for mHealth requires one to solve four problems: securing the data in the sensor nodes, securing the data during communication between the sensors and the mobile phone, securing the data (and processing) inside the mobile phone, and securing the data inside the back-end servers. In this paper we address the second problem; we assume tamper-resistant sensor nodes, and expect the phone platform and the back-end servers to be secured by other solutions. There are many proposed protocols for Wi-Fi networks that provide a secure and private communication [3, 17, and their references], but these protocols are not suitable for low-power sensor networks because of their large transmission overhead. In this paper we propose a protocol (inspired by previous work on privacy-preserving wireless protocols) that provides strong security and privacy properties, and is also energy-efficient, which makes it suitable for low-power mHealth sensing networks.

We make three contributions here:

1. We propose Hide-n-Sense (HnS), a protocol that provides strong security and privacy properties for mHealth sensing with low energy overhead for the sensors. We use three novel techniques in the protocol: i) adaptive security to dynamically change the transmission overhead to maintain security while limiting overhead, ii) MAC striping to make the protocol strongly resistant against forgery for small-sized message authenticate codes (MACs), and iii) asymmetric energy requirements among the communicating devices. HnS is an adaptation of SlyFi [17], where we apply these techniques to provide an energy-efficient protocol for low-power sensor networks, and a granular control over the trade-off between energy and privacy. (We compare HnS with SlyFi in §5 and §7.) Although we demonstrate these techniques using an adaptation of SlyFi, these techniques can be applied to other protocols to make them energy-efficient while maintaining their security and privacy properties.

2. We implemented a prototype of the HnS protocol on TI's eZ430 Chronos wireless device, and we demonstrate experimentally that it is feasible to get strong security and privacy guarantees on low-powered devices.

3. We provide a security, privacy, and energy analysis of our protocol.

We present an overview of our mHealth system architecture and its security model in §2. In §3 we present our protocol, and analyze its security and privacy in §4. We present an evaluation of the protocol in §5, followed by discussion in §6. In §7 we describe related work.

2 Security model

In this section we give an overview of the system architecture, define our threat model and adversary model, identify our security goals, and list trust assumptions.

2.1 System model

Whether for remote-patient monitoring or for personal health management, in a typical mHealth-sensing system the patient carries some sort of *mobile node (MN)*, most likely a mobile phone, that acts as a gateway between a body-area network of *sensor nodes (SN)* and the Internet (as shown in Figure 1). Some SNs collect continuous physiological data about the patient (e.g., heart rate or blood oxygenation). Other SNs may be used intermittently, such as a blood-pressure cuff used daily. The MN may also have useful internal sensors (such as the accelerometers, cameras, or GPS available in many smart phones). The SNs forward sensor data collected at the instruction of the MN, which may process or aggregate the information before presenting it to the patient (locally) or forwarding it via the Internet for use by a remote consumer (health provider, caregiver, researcher) or for their own later use.

Many mHealth sensors are appearing on the market; although most of today's devices use proprietary interfaces, protocols, or gateways, the Continua Alliance has been developing standards and certifying compatible devices [36]. Some of today's devices push their data to a vendor-managed portal where patients view their data [13, for example], but some are capable of uploading to a Personal Health Record (PHR) like Microsoft HealthVault [23]. In this paper, we refer to all of these records, providers, vendors, and other data consumers as a *back-end services*.

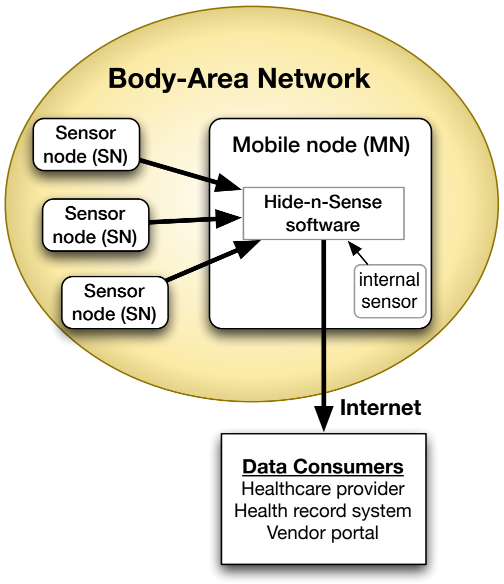

Figure 1: Overall system architecture

As context for our design, and the basis of our security analysis in §4, we first detail our underlying assumptions.

System assumptions. We make the following assumptions about the hardware and software capabilities of the MN and SN .

S1. *Crypto.* Each SN has the cryptographic capabilities needed for message confidentiality and authenticity; that is, each SN has enough resources to support cryptographic primitives such as AES, and SHA-1 (either in software or hardware).

S2. *Clock.* Each SN has an embedded real-time clock that is sufficiently accurate to timestamp its data; the clock's drift can be corrected by synchronizing occasionally with the MN, which is assumed to be synchronized to an Internet-based time source.

S3. *Platform.* The MN is a general-purpose mobile platform, such as a smart phone, with a short-range wireless interface for body-area communication with the SNs (e.g., Bluetooth or Zigbee) and an optional long-range wireless interface for Internet communication (e.g., Wi-Fi or 3G).

S4. *Out-of-band (OOB) channel.* There exists a secure channel (typically an out-of-band channel) between the MN and each SN, which can be used for exchanging secret keys during a pairing process.

2.2 Adversary and threat model

We assume a powerful adversary that has the following capabilities:

A1. *Full access to the wireless channel.* The adversary can *observe, inject, modify* or *disrupt* any message transmitted over the wireless channel between MN and SN .

A2. *No access to out-of-band channel.* The adversary has no access to the out-of-band channel (**S4**) used by the MN, and the SN for the pairing operations.

A3. *Computationally bounded.* The adversary is incapable of breaking cryptographic primitives such as AES and SHA.

A4. *No compromise of MN or SN.* The adversary does not have the capability to compromise either the software or the hardware of the MN or the SN (at least, without it being immediately evident to the patient). We realize that this assumption may be difficult to achieve. We address MN security in another work [33]; as for the

SN one may assume tamper-resistant SN packaging. We focus herein on the problem of securing the wireless communication, so for the purpose of this paper we assume the adversary cannot discover the secret keys (which are used to secure the channel) by compromising the MN or the SN .

A5. *Local adversary.* The adversary does not compromise any of the back-end services. If the adversary has access to the back-end services, he can get access to the patient's sensitive data, defeating the purpose of a secure wireless BASN protocol.

These assumptions begin to define the scope of our solution; in this paper, we focus on the mobile edge to the mHealth ecosystem, and in particular on the communications between the MN and SN . We provide a secure and energy-efficient protocol at the link-layer level; we leave physical-layer attacks to other physical-layer solutions.

Given the capabilities of the adversary, we focus on the following threats. For a more thorough treatment of privacy-related threats to mHealth, see our survey [4] and paper [19].

T1. *Threat to privacy:* The adversary wants to learn sensitive information about the patient, including medical conditions (e.g., disease or treatment type), sensing situation (e.g., types of sensors or number of sensors), or other personal information deemed private (e.g., mobility traces or activity patterns). For this threat, the adversary attempts to eavesdrop on the SN-MN communication, and then discover sensitive information from the messages (if available in cleartext), or by applying traffic-analysis techniques [37].

T2. *Threat to data integrity and authenticity:* The adversary wants to cause the MN to accept incorrect, invalid, or duplicate sensor data. For this threat, the adversary attempts to *forge* a message that looks legitimate to the MN, *tamper* with a legitimate message from the SN, or *replay* a previously observed message.

T3. *Threat of resource exhaustion attack:* The adversary wants to exhaust the MN's battery to prevent the MN from collecting the victim's medical data. For this threat, the adversary may send the MN a series of invalid messages forcing the MN to consume power to receive, process, and discard the messages.

2.3 Security and privacy goals

We must be precise in defining the set of properties that an mHealth-sensing system should achieve; the following set of goals directly address the set of threats defined above. We first list the security properties that are essential to protect security and privacy in MN-SN communications; these properties define our design goals. Unless otherwise specified, the term 'node' refers to both the MN and the SN .

SP1. *Node anonymity.* The protocol should not reveal information about the nodes to an (active or passive) observer (addressing **T1**). To preserve node anonymity, transmissions to/from a node must be *unlinkable*; that is, given two transmissions, the adversary should not be able to tell whether they are to/from the same node.

The HnS protocol preserves the weak unlinkability property of SlyFi; that is, given two messages the adversary cannot tell whether they are from/to a same node using only the content of the messages. To achieve strong unlinkability, the protocol must deal with traffic-analysis attacks, in which the adversary makes use of message size and timing; we leave the problem of finding an energy-efficient defense against traffic analysis for future work.

SP2. *Data confidentiality.* The protocol should not reveal any information about the message content to an observer – active or passive (addressing **T1**).

SP3. *Data integrity and authenticity.* A node should be able to verify that the message it received was generated by the node that claims to have generated it, and that the message was not modified in transit (addressing **T2**).

SP4. *Data freshness.* A node should be able to ignore any duplicate or out-of-order messages. If a node receives such a message, it should discard it (addressing **T2**).

SP5. *Efficient message filtering.* The MN should be able to ignore messages that are not from the SN, quickly and with minimum energy expenditure (addressing **T3**).

2.4 Other properties

To secure the overall system, one also needs other properties; we do not specifically address these properties in our solution because either they are out of scope of this paper or there exist other reasonable solutions in the literature.

OP1. *Node integrity.* The hardware and software of the MN and SN should remain intact, or at least, any unauthorized change should be detectable. SN integrity can be achieved by using tamper-resistant or tamper-evident packaging . To achieve MN integrity, one may leverage trusted hardware [12], hypervisors [16], or other techniques [33].

OP2. *Node confidentiality.* The data within the MN and SN (e.g., secret keys, sensor data, identities of corresponders) should remain confidential to the adversary, either passive or active.

OP3. *Patient Authenticity.* A mobile node should be able to verify that the sensor data it receives was generated by the sensor node located on the correct patient's body, not on some nearby body. (We solve aspects of this problem in other papers [10, 34].)

OP4. *Data availability.* An adversary should not be able to prevent the MN and SN from collecting sensor data, or prevent the MN from delivering the sensor data to the back-end service. Although it is hard to prevent jamming attacks, common measures such as frequency hopping (e.g., used in Bluetooth) make it difficult for the adversary to jam a network.

2.5 Trust model

Any trustworthy system is built on certain assumptions about who trusts whom to do what. We outline our assumptions about the three types of principals – manufacturer, health provider, and patient – in our system. A manufacturer is an entity that produces the sensors or mobile nodes, and distributes them to the health provider and the patient. A patient is a person that uses sensors (obtained either directly from a manufacturer or from a health provider) to get information about his or her own health. The patient can choose to forward this information to a healthcare provider for consultation or use it for self-monitoring. A healthcare provider is an entity that provides health services to the patient, including providing and configuring sensors, monitoring the resulting data, and providing health advice or treatment.

TR1. The patient and the healthcare provider trust the SN manufacturer to produce calibrated sensors that operate correctly, so that the patient and the health provider can trust the sensor to provide the right reading.

TR2. The patient and the healthcare provider trust the MN and HnS manufacturers to write correct software; thus, they protect confidentiality and integrity according to the above goals.

TR3. The patient trusts the healthcare provider not to disclose the sensor data to unauthorized parties.

TR4. The healthcare provider trusts the patient to not be malicious; thus, the patient does not tamper with the hardware or software of the mobile node or the sensor node.

The manufacturer has no stake in the system, so the manufacturer assumes nothing about other principals.

3 Hide-n-Sense

We begin with a brief use case of a patient using the Hide-n-Sense (HnS) protocol to secure communication of her body-area sensor network (BASN). The patient obtains a new SN and wishes to use it with her MN. She *pairs* the SN and the MN, as described below, allowing them to authenticate each other and to share the keys used for discovery. Later, when she puts on the SN to use it, the SN automatically discovers the MN (which she routinely carries) using the HnS discovery protocol, and the two nodes establish a secure session. Then the SN securely uploads the sensed data to the MN, using the HnS session protocol, until the patient removes her SN.

The HnS protocol presented here supports a BASN with one MN, and one or more personal SNs, while maintaining the security and privacy properties (as described in §2.3) with a small energy overhead. For simplicity of exposition we describe a BASN with one MN and one SN.

The HnS protocol is a link-layer protocol, and it works in conjunction with a medium-access control protocol that controls access to the wireless channel. That is, HnS decides what needs to be transmitted (e.g., data or ACK contents), and sends the message to the medium-access layer to transmit. The medium-access layer transmits the message in the medium when it can (e.g., in a CCA type protocol, the medium-access layer transmits when the medium is free). To achieve strong privacy, the HnS needs to have full control over the contents of each transmission, so HnS cannot run on top of other protocols such as ZigBee or Wi-Fi. We implemented a HnS prototype on embedded devices that allow us to write our own link-layer protocol (see §5).

3.1 Pairing

Pairing is the process of bootstrapping a secure communication channel between two previously unassociated devices [20]; that is, in pairing two devices share keys, which are used to secure their communication. For instance, when a patient wants to add a new sensor to her BASN, she will pair the new SN to her MN, using a suitable out-of-band (OOB) channel.

A common (but weak) pairing method for Bluetooth devices is the PIN-based pairing method. Some other pairing methods use the mobile phone camera [22] or an accelerometer embedded in devices [6, 21]. For details on various pairing methods we refer the reader to Kumar et al. [20]. Recently Gollakota et al. [15] proposed an in-band pairing method that does not require an OOB channel. Whatever the pairing method, we require it to have the following properties:

Intentionality: The patient can effect pairing between her MN and a new SN when, and only when, she intends to do so, and the pairing results in the intended SN being paired with the intended MN. Intentionality is important because we do not want the patient's MN to be pairing with any device without her knowledge.

Isolation: The adversary cannot insert himself between MN and SN (man-in-the-middle attack).

Secrecy: The adversary cannot observe the key material shared between MN and SN.

In HnS, during pairing, the MN shares five things with the SN: three discovery keys (k_d^h, k_d^p, k_d^m), a threshold for consecutive failed discovery attempts (w_d), and a nonce (N).

3.2 HnS discovery

Once two nodes are paired, they do not necessarily remain in radio contact. To communicate, the nodes must first detect that they are in radio range, and for that purpose they use the discovery protocol. Most discovery protocols, however, give away the identity of the seeker (or the nodes being sought), violating **SP1**.

The HnS discovery protocol is an adaptation of *tryst* in the SlyFi protocol, proposed by Greenstein et al. [17] for Wi-Fi setting. In tryst, the nodes need to frequently update their address table (which they use to filter incoming messages), even if there is no communication underway. Although these operations require negligible resources for powerful devices (such as laptops), for small mobile devices this requirement would drain too much energy; HnS removes the need to frequently update address tables, and it further reduces the burden on the SN, which has limited resources. This asymmetric resource requirement design works well when the MN is a smartphone that may be recharged daily, and the SNs are resource-constrained devices that are inconvenient to recharge.

By design, we allow only the SN to initiate discovery. There are two reasons for this choice. First, the SN does not have to leave its radio on in anticipation of receiving a discovery message from the MN; it can turn off its radio when not in use, and save energy. Second, to filter incoming messages, we show below that a node must maintain a hash table of expected headers; if the SN does not anticipate receiving a message (without it having initiated the communication), the SN does not have to maintain any hash table, and thus, it can save memory.

Figure 2 shows the discovery format of the i^{th} discovery message. The header, payload, and message authentication code (MAC) for the i^{th} discovery message are generated using

$$\begin{aligned} \mathcal{H}_d(i) &= \text{AES}_{k_d^h}(N+i) \\ Payload &= \text{AES-CTR}_{i,k_d^p}(data) \\ \mathcal{M}_d(i) &= \text{AES-CMAC}_{k_d^m}(\mathcal{H}_d(i)||Payload) \end{aligned} \tag{1}$$

where $\mathcal{H}_d$ and $\mathcal{M}_d$ represent the header and MAC of a discovery message (note the subscript d, used to represent discovery variables); i is the discovery message number; k_d^h, k_d^p, k_d^m are the keys shared during pairing (note the superscripts h, p, m; k_d^h is used to encrypt the header, k_d^p is used to encrypt the payload, and k_d^m to generate the MAC); and N is the nonce, also shared during pairing. Note that the discovery message number i is used as nonce in the CTR mode to generate the key stream that is used encrypt the payload; that is, the first payload block is encrypted with the key generated using $\text{AES}_{k_d^p}(i||1)$, the second block is encrypted with the key generated using $\text{AES}_{k_d^p}(i||2)$, and so on.

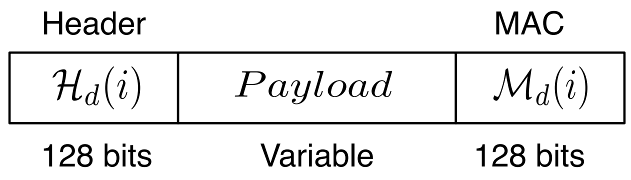

Figure 2: Format of the i^{th} discovery message

The MN maintains a hash table with w_d future discovery headers; that is, it maintains headers from $\mathcal{H}_d(i+1)$ to $\mathcal{H}_d(i+w_d)$. The MN filters incoming messages (i.e., determines whether the receives message was sent by the SN) by doing a hash-table lookup on the header of the received message.

The HnS discovery protocol works as follows:

1. When the SN receives a discovery trigger (described below), the SN uses Equation 1 to generate and send a discovery message, say the i^{th} discovery message. The data in the payload does not matter here; it can be application-related data (e.g., sensor status) or it may be empty.

2. MN replies with the $(i+1)^{th}$ discovery message. The payload contains session keys (k_s^h, k_s^p, k_s^m, k_s^x), session message-loss threshold (w_s), and the length of header and MAC (h_0, m_0) for session messages. The MN generates new session keys for every discovery message from the SN; w_s and (h_0, m_0) are same for all SNs in a BASN, and are set by the MN.

   The MN updates its hash table: it removes old headers (i.e., headers up through $i+1$) from the hash table, and adds future discovery message headers, such that the number of headers in the address table remains w_d; the MN also adds w_s future session headers and removes all session headers from an old session, if any. At this point, a secure session is established from the MN's perspective.

3. After sending a discovery message, the SN waits for incoming messages. If the received message is from the MN (i.e., the message header matches $\mathcal{H}_d(i+1)$) and if the message is authentic (i.e., MAC verification is successful), the SN decrypts the payload, and saves the data (i.e., keys and other parameters). At this point, a secure session is established from the SN's perspective. It records internally the number $i+2$ for use in a future discovery message.

   However, if the SN does not receive any reply from the MN, for a pre-determined period, it gives up and turns off its radio. The SN may retry few times before giving up.

Discovery triggers. A *discovery trigger* tells a node when to initiate the discovery protocol. It is desirable to have a discovery trigger with low delay; that is, the discovery trigger should fire when the application/user expects the discovery to happen (e.g., when a patient puts on the SN or when the SN is near the MN). Discovery protocols in many wireless networks (including Wi-Fi and Bluetooth) use a time-based trigger; that is, nodes periodically send discovery probes to detect the presence of any previously associated (paired) device. Such periodic scanning methods drain too much energy, and the excess transmissions can pose privacy risks. Thus, it is important to choose a trigger for the discovery protocol that achieves quick discovery with few transmissions. Some examples follow.

Physical trigger: The SN can be designed to detect when it is put on by the user (e.g., with a clasp that triggers a switch inside the sensor) or when it is being used (e.g., when a user steps on a weight scale). Once the SN is triggered, it can initiate the discovery protocol by sending a discovery message to probe for the presence of the mobile node that the user should be carrying.

Acceleration-based trigger: If the SN has an embedded accelerometer, which is inexpensive and easy to include in many devices, the SN can recognize when it is touched or moved and use that event to initiate discovery. Thus, an SN left at home on the bedside table will not consume energy looking for its MN, until the patient picks it up to put it on.

Time-based trigger: Although the above two methods are preferred, there are cases or sensor types where those methods will not work. With a time-based trigger, the sensor node will periodically initiate the discovery protocol. This method may require more energy and pose greater privacy risk.

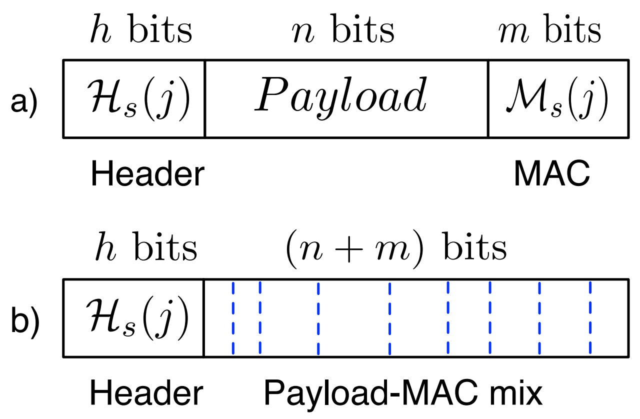

Figure 3: Session message format: a) Before MAC striping, b) After MAC striping (the dashed lines represent the m MAC bits)

3.3 HnS session protocol

The HnS session protocol is used to secure the sensor data that the SN sends to the MN. A session begins after a successful conclusion of the discovery protocol, and it ends if the SN loses w_s consecutive session messages (a message is considered lost if the SN does not receive an ACK for it from the MN, within the ACK timeout period, as defined by the underlying medium-access control protocol).

In the session protocol (as in the discovery protocol) we allow only the SN to initiate the communication (again, to save energy and memory at the SN). This design works well for a BASN, where most of the communication is from the SN to the MN (i.e., sensor data), with the MN occasionally sending control commands to the SN. The message filtering mechanism at the MN is similar to the discovery protocol: the MN maintains a hash table with expected headers for messages from the SN, and it filters incoming messages with a hash-table lookup. A hit in the hash table tells the MN that the message is from the SN, and also from which SN, in case there are multiple SNs in a BASN.

Figure 3 shows the session message format. The header, payload, and MAC for the j^{th} session message are generated using

$$\begin{aligned}
\mathcal{H}(j) &= \text{AES}_{k_s^h}(j) \\
Payload &= \text{AES-CTR}_{j,k_s^p}(data) \\
\mathcal{M}(j) &= \text{AES-CMAC}_{k_s^m}(\mathcal{H}(j)||Payload) \\
\mathcal{H}_s(j) &= \text{MSB}_h(\mathcal{H}(j))\ ;\ \ \mathcal{M}_s(j) = \text{MSB}_m(\mathcal{M}(j))
\end{aligned} \tag{2}$$

where $\mathcal{H}_s$ and $\mathcal{M}_s$ represent the header and MAC of a session message; k_s^h, k_s^p, k_s^m are the session keys shared during discovery; j is the session message number, and (h, m) represent the length of header and MAC (at the beginning of a session, $h = h_0, m = m_0$, where (h_0, m_0) were shared during discovery). The session message number j is also used as a nonce for generating the key stream in the CTR mode to encrypt the payload; thus, the first payload block is encrypted with the key $\text{AES}_{k_s^p}(j||1)$, the second block is encrypted with the key $\text{AES}_{k_s^p}(j||2)$, and so on. Note that the actual bits sent as header and MAC are the h and m most significant bits (MSB) of $\mathcal{H}$ and $\mathcal{M}$, respectively (as shown in Figure 3a). The actual session message that is transmitted is shown in Figure 3b, which is generated after MAC striping the message shown in Figure 3a; we describe the MAC striping technique next.

Unlike discovery messages, session messages are transmitted frequently, so we wish to keep the session message overhead (size of header and MAC) as small as possible. However, reducing the size of header and MAC also decreases the protocol's resistance to forgery. So, to reduce the message overhead while maintaining reasonable security, we propose two techniques: *MAC striping* (to make the session protocol resistant against selective forgery, even for small MAC sizes), and *adaptive security* (to dynamically change message overhead to maintain security against existential forgery).

3.3.1 MAC striping

MAC striping provides strong resistance against selective forgery. In selective forgery the adversary tries to forge a message with the payload of his choice. Thus, for the adversary to forge a message of format shown in Figure 3a, with a ciphertext payload of his choice, he requires work proportional to 2^{h+m} (he needs to guess the correct h-bit header and m-bit MAC). However, a clever adversary may intercept[1] a message (**A1**) to discover a valid header, and thereby reduce the work to 2^m. MAC striping changes the message format slightly (as shown in Figure 3b), which makes the protocol strongly resistant against selective forgery, even with a small MAC. The MAC bits are interspersed in the payload at different offsets, represented by dashed lines in Figure 3b. These bit locations are different for each message, and are generated by a pseudorandom sequence generator function f (HnS uses AES):

$$\langle x_0, x_1, ..., x_m \rangle = f(k_s^x, j, n, m), \tag{3}$$

where x_i ($< n$) is the offset for the i^{th} most significant bit of $\mathcal{M}_s(j)$, k_s^x is the key that was shared during discovery, j is the session message number, n is size of the payload, and m is size of the MAC. Note that these offsets are computed by the MN and the SN independently.

When a node (SN or MN) receives a session message (it will be of format Figure 3b) with a valid header, the node computes the pseudorandom sequence $\langle x_0, x_1, \ldots, x_m \rangle$, and separates the MAC bits from the payload (to recover a message of the format in Figure 3a).

The advantage of MAC striping is that even with small sized MAC (i.e., small m) it provides strong resistance against selective forgery; we explain this property in detail in §4.

3.3.2 Adaptive security

In wireless protocols the receiver node uses the header to filter incoming messages (that is, to determine whether the incoming message is addressed to this node), and it uses the MAC $\mathcal{M}$ to verify whether the received message is authentic. The overhead (header and MAC) in these protocols is usually fixed, and for strong security, the protocols choose long header and long MAC. However, a node is not always in a hostile environment, so using large overhead all the time is inefficient. In many mobile devices, the transmission is expensive (energy-wise), so a low-power sensor network should have a small transmission overhead.

Adaptive security provides strong security when required (e.g., when a network is under attack), but saves energy, at both the sender node and the receiver node, when strong security is not required. To achieve this, the nodes dynamically change the size of the header and the MAC sent in the message. That is, instead of using a fixed large overhead, the nodes use a small message overhead that expands dynamically if the node detects the presence of an adversary who is trying to forge a message.

Adaptive security: How to adapt. Consider a simple body area sensor network (BASN) with a sensor node (SN), and a mobile node (MN, e.g., a smartphone). The MN chooses the header and MAC sizes to be used by all the SNs in the BASN. When the MN decides that it needs to change either the header or the MAC size (see below for a discussion on how), it notifies the SN by sending a *reissue* message. During this adaptive process, the MN ensures that communication is not disrupted due to inconsistency in message overhead sizes.

The MN maintains a set σ_{MN} with pairs representing sizes of header and MAC: one pair representing the old header and MAC sizes (currently being used by the SN), and the second pair representing new header and MAC sizes that the MN wants the SN to use. When the MN receives a message, it parses the message to get the header using the new header size from σ_{MN}. If the message header is valid, but the message MAC is wrong, the MN will parse the same message with the old header size (if any) from σ_{MN}. This ensures that the MN can receive messages sent by the SN using either the old or new values for header and MAC sizes. Once the MN knows that the SN has successfully adapted to the new header and MAC sizes (i.e., when it receives a message from SN with the new header and MAC sizes), it removes the old pair from σ_{MN}.

Figure 4 shows how the MN and the SN adapt their header and MAC sizes from (h_0, m_0) to (h_1, m_1). In the figure we show a set σ_{SN} for the SN; this set will always contain one pair of values representing the header and MAC sizes that the SN will use to parse incoming messages, and to send messages to the MN. Thus, the MN will be able to receive a message from the SN if $\sigma_{SN} \subseteq \sigma_{MN}$.

[1]To intercept a message, the adversary captures the message header when it is being transmitted, and then disrupts some bits in the payload or the MAC so that the receiver discards the message because it will fail the MAC verification process.

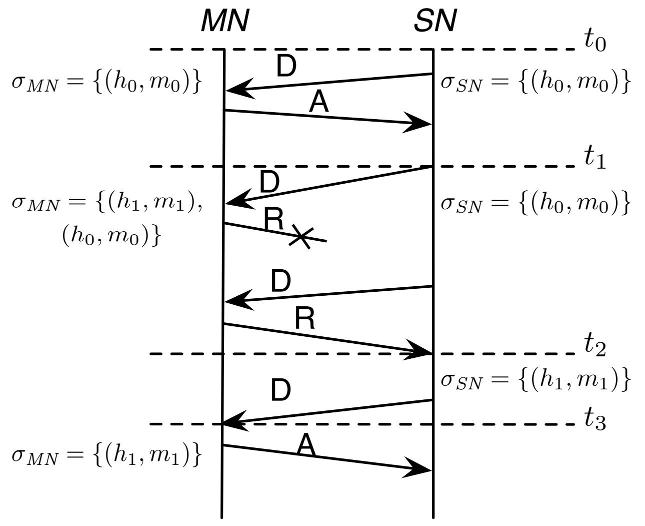
MN
SN
t_0
t_1
t_2
t_3
D
A
R
$\sigma_{MN} = \{(h_0, m_0)\}$
$\sigma_{SN} = \{(h_0, m_0)\}$
$\sigma_{MN} = \{(h_1, m_1), (h_0, m_0)\}$
$\sigma_{SN} = \{(h_0, m_0)\}$
$\sigma_{SN} = \{(h_1, m_1)\}$
$\sigma_{MN} = \{(h_1, m_1)\}$

Figure 4: Adaptive security at work (D = data, A = ack, R = reissue)

As shown in Figure 4, at time t_0 (i.e., during discovery), the MN shares the values (h_0, m_0) with the SN. Later, at time t_1, the MN wants to change the message overhead size to (h_1, m_1); it adds (h_1, m_1) to σ_{MN}. After time t_1, when the MN receives a message from the SN (the overhead for this message will be of length (h_0, m_0)), it receives the message (because $(h_0, m_0) \in \sigma_{MN}$), but it does not accept it. The MN replies back to the SN with a reissue command message that tells the SN to start using (h_1, m_1). However, the SN may not receive the response (as shown in the figure), in which case the SN will continue to send messages with overhead of size (h_0, m_0), which is okay because the MN can receive those messages as $(h_0, m_0) \in \sigma_{MN}$. Eventually, at time t_2, the SN receives the reissue message, and replaces (h_0, m_0) in σ_{SN} with (h_1, m_1). At time t_3, when the MN receives a message from SN, it knows that the SN has updated its σ_{SN}, and so the MN will remove the old values (h_0, m_0) from the σ_{MN}. In a BASN with more than one SN, the set σ_{MN} may contain more than two elements, depending on how quickly SNs adapt.

Adaptive security: When to adapt. The MN decides when the BASN must adapt to new header and MAC sizes; the decision on *when* to adapt can be based on many factors, such as application requirements, desired security against forgery, and network bandwidth optimization. An in-depth analysis of these factors is outside the scope of this paper, but we suggest below an example, and use it in our experiment.

The MN uses the header to filter incoming messages. When the header of the incoming message is valid (i.e., the header matches with one of the headers in the MN's address table), the MN does MAC verification to determine whether the message is authentic. If the MAC verification fails, it implies that either the incoming message was from the SN but got corrupted in transit, or that the message is of a neighboring BASN that happened to use a header considered valid by the MN, or that the message is a forgery attempt. It is hard for the MN to distinguish between the three cases. We take a cautious approach, and consider a failed MAC verification to be a forgery attempt.

In adaptive security MN dynamically changes the header and MAC sizes (to save energy) while maintaining a certain level of security. We can define this certain level of security as: for a set of any w_s headers in the table, the probability that an adversary will successfully forge a message with any of those headers should always be below β, where β is chosen by the application/patient.

Let T be a time period and ρ be the probability of successful forgery during that time period T. The SN's lifespan can be considered as a series of time periods. Thus, to maintain the security guarantee we get the condition:

$$\beta \geq 1 - (1 - \rho)^n \tag{4}$$

where n is the number of time periods in the lifespan of the node; that is, lifespan of the sensor = $\sum_{i=1}^{n} T_i$.

For a given T and ρ, the HnS adaptive security method maintains the successful forgery probability for this period less than ρ irrespective of the number of forgery attempts by an adversary.

The MN tracks the number of failed MAC verifications. The probability that one failed MAC verification would succeed (that is, probability of a successful forgery given that the adversary guessed the header correctly) is 2^{-m}, where m is the MAC size. Thus, the probability of successful forgery in x failed MAC verifications is

$$P = 1 - \left(1 - 2^{-m}\right)^{x} \tag{5}$$

We want this probability to be less than ρ. Thus, solving $\rho \geq P$ for x we get

$$a \leq \frac{\log\left(1-\rho\right)}{\log\left(1-2^{-m}\right)} \tag{6}$$

Whenever the number of failed MAC verifications exceeds $\frac{\log(1-\rho)}{\log(1-2^{-m})}$, the MN will first increase the header size, and then the MAC size. Increasing the header size will reduce the rate of failed MAC verifications if the source of those failed MAC verifications was the neighbouring BASN traffic or an adversary trying to randomly guess the message. However, a clever adversary can intercept messages to get the headers, and then try to forge a message by attaching different payload-MAC values. Increasing the header size will not help against such an adversary. So against such an adversary the MN increases the MAC size m (by observing the number of transmission attempts and number of failed MAC verifications, the MN can guess if the failed MAC verification was due to random/exhaustive header guessing or if the adversary knows the header). Once x exceeds the threshold (Equation 6), the MN does not accept any messages, but if the message is valid, the MN sends a reissue command to indicate the change in header/MAC size.

The MN falls back to smaller header and MAC sizes after the time period T expires. The MN can choose to keep T and ρ constant, but it can optimize by initially choosing ρ value close to β and then varying ρ and T for future time periods, to maintain the condition in Equation 4 throughout the lifespan of the sensor.

4 Security and privacy analysis

SlyFi achieves the properties listed in §2.3. HnS is an energy-efficient adaptation of SlyFi. It is important that the security and privacy properties are not compromised to save energy. In this section we explain how HnS preserves/achieves these properties.

Node anonymity (SP1). SlyFi achieves this property by achieving weakly unlinkability; that is, the adversary cannot link multiple messages as coming from/to the same node by observing the message contents. To do so, it encrypts both the header and payload, and ensures that each message appears as a pseudorandom string to the adversary.

The HnS discovery message format is similar to SlyFi, so the discovery messages are unlinkable. But the HnS session message format is different. In HnS session protocol, the header and the MAC are truncated. Since the original header and original MAC (i.e., the full 128-bit header $\mathcal{H}(j)$ and MAC $\mathcal{M}(j)$ generated using Equation 2) are AES ciphertext output, their truncated strings are also pseudorandom strings. Since MAC striping is a pseudorandom rearrangement of bits, and given that the original string was pseudorandom, the rearrangement after MAC striping is also pseudorandom, for the adversary. Thus, the session message (as shown in Figure 3) appears as a pseudorandom string to an adversary, and the adversary cannot link the session messages. Thus, HnS preserves the weak unlinkability property.

Data confidentiality (SP2). The HnS protocol achieves data confidentiality by using AES (a standard stream cipher) in counter mode (CTR) to encrypt the payload. To avoid the adversary identifying two messages having the same payload, HnS generates different ciphertexts even for the same payload by using the message number as a nonce for generating the key stream to encrypt the payload. That is, the key stream to encrypt the payload of i^{th} message is generated using $\text{AES}_k(i||b)$, where b is the payload block number, and k is k_d^p for discovery messages and k_s^p for session messages.

Data integrity and authenticity (SP3). HnS prevents unauthorized change to the header or payload by use of a message authentication code (MAC). Nonetheless, the adversary may attempt to alter the message content or construct a new message without being detected; such an attack is called *forgery attack*. We consider two types of forgery attacks: *selective forgery* (when the adversary tries to get the receiver to accept a message with a payload of his choice), and *existential forgery* (when the adversary tries to get the receiver to accept a message with any payload).

Selective forgery: In selective forgery the adversary chooses a payload, which is a ciphertext (Equation 1 and 2), and tries to find the corresponding MAC.[2] Since the adversary does not know the MAC key (shared during discovery), the adversary picks a random MAC out of $\{0,1\}^m$, where m is the length of the MAC. To successfully forge a message the adversary needs to get the right header and right MAC for a given payload. We assume, however, that the adversary can obtain the right header (by intercepting messages). So for a successful selective forgery the adversary only needs to guess the right MAC for a payload of his choice.

The HnS discovery messages use a 128-bit MAC, so to selectively forge a discovery message, the adversary would need work proportional to 2^{128}. The HnS session protocol, however, uses a small-sized MAC. Without MAC striping (see message format in Figure 3a), the probability that the attacker succeeds in selectively forging a session message with one random guess is 2^{-m}. With MAC striping (see message format in Figure 3b), the MAC bits are interspersed with the payload bits at locations determined by Equation 3. Without knowing the key k_x^s and the message number j, it is computationally hard for the adversary to know which bits among the $(n+m)$ bits are the MAC bits, where n is the size of the payload. Therefore, to forge a message of his choice, the adversary has to guess the matching MAC bits out of 2^m possibilities, and also MAC-bit locations out of $\binom{n+m}{m}$ possible MAC-bit locations, making the probability of success $\frac{1}{2^m\binom{n+m}{m}}$. For example, when the payload is 10 bytes long and the MAC is 2 bytes long (i.e., $n = 80, m = 16$), the success probability of selective forgery without MAC striping is 2^{-16}, and with MAC striping it becomes approximately 2^{-75} (since $\binom{96}{16} \approx 2^{59}$). Therefore, the MAC striping technique drastically decreases the success probability of selective forgery (from 2^{-16} to 2^{-75} in the example) without increasing the MAC size.

Existential forgery: In existential forgery the adversary tries to find any matching payload-MAC pair that the MN would accept; the adversary has no control over the payload content. Without knowing the MAC key, the adversary can only guess, choosing an $(n+m)$-bits long $payload||MAC$ string out of $\{0,1\}^{n+m}$. Out of 2^{n+m} possible such strings, only 2^n payloads exist, thus the success probability of such a random guess is 2^{-m}.

The HnS discovery protocol uses 128-bit MAC. Thus, the work required for existentially forging a discovery message is also proportional to 2^{128}. For the HnS session protocol, the work required for existential forgery is the same with and without MAC-striping if the guess is made uniformly at random. Thus, to existentially forge a session message the adversary would need work proportional to 2^m. The HnS session protocol uses a short-size MAC, but it uses adaptive security to ensure that the probability of a successful existential forgery is never greater than β (as described in §3.3.2).

Data freshness (SP4). Each message in the HnS protocol has an encrypted header that contains the message sequence number. Recall that the MN maintains a hash table of headers of expected messages, and when it receives a message from the SN it removes all the headers with the sequence number less than the received message from the hash table. Thus, the MN will only accept messages in increasing order of their sequence number, achieving **SP4**.

Efficient message filtering (SP5). If the MN requires lot of processing to discard invalid messages, an adversary can launch a resource-exhaustion attack, where the adversary will try to drain the MN's battery by sending a stream of invalid messages. HnS uses the message filtering method proposed in SlyFi. The MN uses hash-table lookups to filter incoming messages, and it discards invalid messages (i.e., message with headers that are not available in the hash-table). For any message that has a matching header in the hash table, the MN does MAC verification, and discards the message if the MAC verification fails. Both the operations (hash-table lookup and MAC verification) require negligible energy and processing time on the MN. Thus, the message filtering mechanism is efficient enough to thwart resource-exhaustion attacks.

5 Evaluation

In this section we describe the implementation of HnS, and its evaluation. We experimentally simulate an adversary and measure how the HnS adapts to forgery attempts of the adversary. We also compare energy cost of HnS with SlyFi and SPINS [26] (a security protocol for low-power wireless sensor networks).

[2]For the adversary to inject sensor data chosen by itself, the adversary needs to compute the corresponding ciphertext, which is difficult because it requires knowledge of the encryption key and nonce. As a forgery attack, however, it suffices to make the MN accept the ciphertext chosen by the adversary, whatever the decrypted data might be.

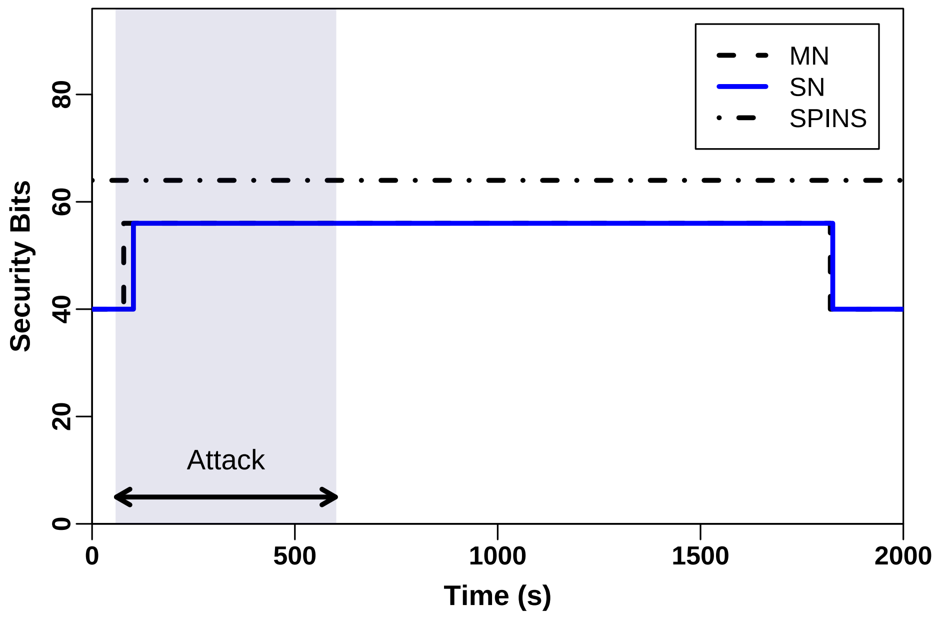

Figure 5: MN and SN adapting in response to an attack.

To evaluate the efficacy of HnS, we implemented HnS on the eZ430 Chronos wireless wrist device [9] from Texas Instruments, which integrates a 16-bit ultra low-power MCU (MSP430), a CC1101 wireless transceiver, 32KB flash and 4KB RAM into a single chip. The platform also features in-hardware AES-128 encryption, which we use in HnS. The device includes three integrated sensors (3-axis accelerometer, pressure, and temperature), that we use as a proxy for mHealth sensors in our experiments.

Using this hardware setup, our experiments focus first on measuring HnS's ability to respond to an attack, and second on the impact that using HnS has on the energy consumption of the system.

5.1 Adaptive security

In our first experiment we use three Chronos watches, acting as an SN, an MN and an adversary. We use a watch for the MN, instead of a phone, because the Chronos allows us to implement HnS at the link layer. The MN and SN use the HnS protocol, and the adversary tries to forge a message. The goal of the experiment is to observe how the MN adapts to forgery attempts, and how the SN and MN coordinate the change in the number of security bits used.

Experiment setup: The SN imitates a temperature sensor that sends a reading every 15 seconds; the sensor payload is 6 bytes. We assume a strong attacker, who knows the headers in MN's table. The adversary attempts to forge messages for 10 minutes at a rate of 100 forgery attempts per second (this rate saturates the wireless channel), after which communication returns to normal.

Figure 5 shows the result of this experiment. The number of security bits used is shown over time for the MN, then SN, and SPINS [26]. SPINS is a security protocol for wireless sensors, and we use SPINS as a benchmark for comparison. SPINS uses a 64-bit MAC as protection against message forgery. When not under attack, HnS achieved much lower overhead than SPINS. During the attack, HnS increased its overhead; however, the overhead never needed to be raised to SPINS's level.

During the experiment, both the MN and SN start using a 2-byte header and 3-byte MAC (total overhead= 40 bits). The MN chooses $\beta = 2^{-24}, \rho = 2^{-16}$ and $T = 30$ min. Beginning at time $t = 60$ the attacker transmits messages with random payload-MAC bits but using a header it knows is in the MN's hash table. Such a strong attacker only has to transmit 256 messages to force the MN to increase its MAC size m (see Equation 6). After the MN increases m to 32 bits, the attacker must send 2^{16} messages before m increases again, but the attack ended before that. After the attack has ended, the MN and SN both slowly return to their previous security level after time T.

In this experiment, the attack was limited to 10 minutes. Figure 6 shows how HnS adapts in the presence of

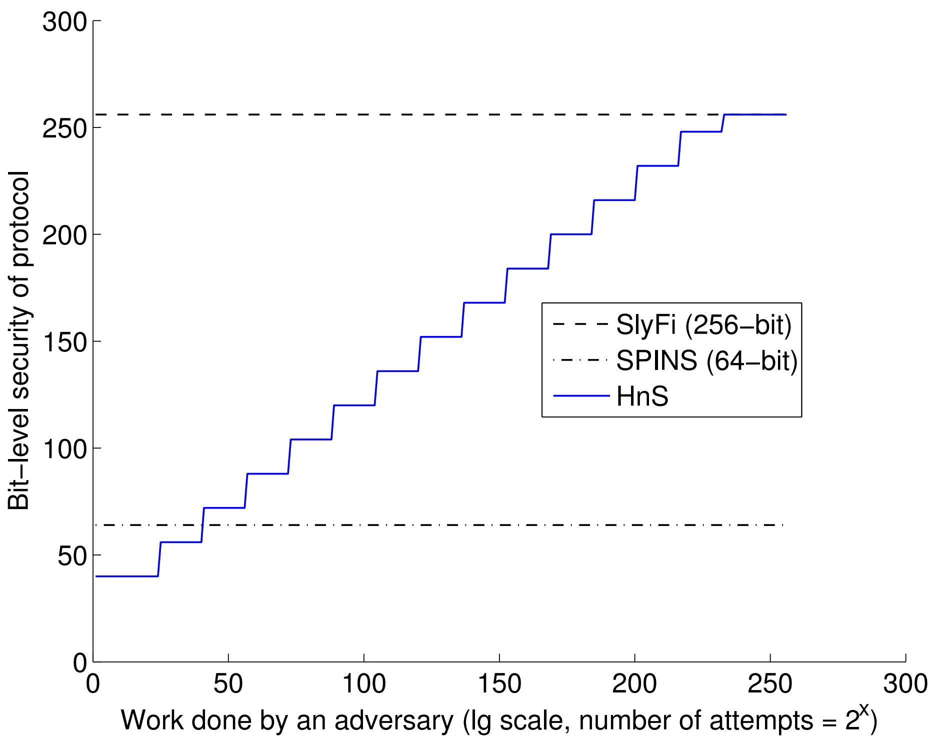

Figure 6: Adaptive security against forgery attack

Table 1: Comparing energy cost of three protocols

	SlyFi	SPINS	HnS
Transmissions overhead	256-bit	64-bit	variable (up to 256 bits)
Security and Privacy[1]	Yes	No	Yes
Average power (mW)	27.60	14.74	13.11
Battery life (hr)	25.49	47.74	53.68

[1]That is, whether the protocol achieves the security and privacy goals described in §2.3.

a persistent attack. In this unlikely case, the MN will continue to increase the security bits used in response to an increasing number of forgery attempts (log-scale x-axis). The y-axis represents the security bits used by the protocol's header and MAC (i.e., $h + m$). In the plot, the initial overhead is set to 48 bits, and the probability that a forgery succeeds is set to 2^{-16} (i.e., $\rho = 2^{-16}$). The plot also compares HnS's overhead to that of the SlyFi (256-bit) and SPINS (64-bit) protocols.

5.2 Energy analysis

In our second experiment, we measured the energy required to run a SlyFi-like privacy-preserving wireless protocol, SPINS, and HnS. We simulated the traffic produced by a six-electrode ECG sensor, which sends 10-byte messages (a timestamp and 1 byte per electrode) at a rate of 50 Hz. We measured the energy consumed using a Monsoon power monitor [24]. Table 1 presents the comparison of the three protocols.

Since communication cost dominated the energy consumption, among the three protocols, the HnS protocol had the least overhead and hence it consumed less power and had greater battery life. Compared to SlyFi, HnS provides similar security and privacy properties at about half the energy cost, and compared to SPINS, HnS provides more security and privacy properties, and extends the sensor life by about 12%.

6 Discussion

Attacks on SN In the sections above we described only attacks on the MN because the protocol only allows a small attack window for the SN; the MN never initiates communication and thus the SN rarely listens for messages. The SN's radio is on only when it is waiting for an ACK, seeking for only one particular header in the incoming message (unlike the MN who has to match the header of incoming message with all the headers from the hash table). Thus, the adversary will have one chance to forge an ACK, and the success probability for that is $2^{-(h+m)}$.

Energy cost at MN In the preceding sections we focus on energy consumption in the low-power SNs; energy consumption by the MN may also be a concern. Given the patient's desire to use mHealth sensing, however, HnS makes power consumption at the MN no worse than it would be in a comparable low-power security protocol (e.g., SPINS) and indeed much less than in a privacy-preserving wireless protocol like SlyFi.

Selective vs. existential forgery. In the context of mHealth sensing it should be easy for the MN to detect existential forgery attempts, because any message accepted and decrypted will produce arbitrary sensor data values, because the adversary has no control over the ciphertext payload. These values will likely be discarded due to semantic checks on the format or reasonable value range for such data. In case of selective forgery, the adversary can choose the ciphertext payload. Although the adversary cannot choose the underlying sensor data because he does not know the encryption key, he can use the previously observed ciphertexts to generate (or choose) a ciphertext that will give a valid sensor data when the MN decrypts the ciphertext. Thus, it is important to have strong resistance against selective forgery in mHealth sensing, as provided by HnS.

Limitations HnS is a link-layer protocol. It does not provide any defense at the PHY layer. An adversary may use PHY-layer fingerprinting to link together a set of transmissions to/from a node, breaking the unlinkability property. However, such attacks require special hardware, raising the bar for the adversary [25].

HnS does not yet defend against traffic-analysis attacks. HnS (like SlyFi) makes traffic analysis more difficult by removing all explicit identifiers from a message, so that the analyst must first link messages before conducting analysis of message content. Still, an adversary may be able to use message size and message timing to link transmissions from/to a node. The development of an energy-efficient solution against traffic analysis is outside of the scope of this paper; we leave it for future work.

7 Related work

Several research projects have proposed, and even demonstrated, prototypes for mobile healthcare sensing, but providing a comprehensive security and privacy solution that is energy-efficient for low-power sensors is still an open question. The HnS protocol that we present in this paper is part of the HnS architecture, which is our effort towards building such a complete solution.

Shnayder et al. [30] developed CodeBlue, a hardware and software platform for mHealth sensing by integrating medical sensors with wireless motes. The focus of their work was on demonstrating the feasibility and performance of such an mHealth sensing system in terms of scalability, channel fairness, latency and jitter, effects of mobility, and message loss. Although the authors acknowledge the need for security and privacy for mHealth sensing, they do not address it in their work.

Garcia-Morchon et al. [14] propose a development model and security framework for a mHealth sensing system in a hospital setting; the patient's caregiver can access the sensor information from the patient's BASN. Their security and privacy goals for a user's BASN are similar to our goals, but their protocol does not achieve all of those goals: to preserve user identity, they use pseudonyms for identifiers in transmissions [5], but pseudonyms do not provide strong user privacy because an adversary can link transmissions by correlating the old and new pseudonyms based on the movement pattern of the user [18]. In HnS, we use a technique from SlyFi [17] where identifiers in the message change with each transmission, and hence are much more difficult to correlate. Furthermore, in their model they focus on improving energy efficiency by using lightweight encryption algorithms, which is beneficial, but the energy savings obtained by reducing the communication overhead (as in HnS) are more substantial. (On embedded platforms like the SN, radio communication is often an order of magnitude more expensive than computation, especially if encryption hardware is included.) Third, they do not provide an energy cost of their model (i.e., energy consumed by a sensor if it were to use their protocol), whereas in HnS we provide energy measurements for HnS protocol, and show it to be much more efficient that other protocols.

Privacy-preserving wireless protocols The class of privacy-preserving wireless protocols (PPWP) represent all the protocols that try to provide user privacy in a wireless network, by obfuscating any information in the transmission that an adversary can use to get some information about the user (or her device). There are several PPWP proposed in the literature [3, 17, 32, and their references]. We use some techniques from SlyFi [17] (a relatively strong PPWP), and use it as a basis for comparison while evaluating the HnS protocol. Unlike SlyFi, our focus is to provide strong privacy and security with low energy overhead so that it can be applied to low-power sensors in a BASN. Hence, HnS differs in the following ways: i) in HnS we made the resource requirement asymmetric between the two nodes involved in the protocol (that is, one node requires less resources than the other), so that even a lower-capability SN can participate without undue energy consumption; ii) HnS uses smaller header and MAC sizes, which improves energy efficiency and network bandwidth, but to maintain strong security we use an adaptive security model and MAC striping; iii) we provide an energy evaluation for the protocol; iv) in HnS the nodes do not have to rely on synchronized clocks for discovery, reducing complexity.

Perrig et al. [26] proposed the SPINS security protocol for low-power wireless sensors. They did not consider node privacy, however. To achieve data integrity and authentication, they propose using a constant-length 8-byte MAC. HnS uses a smaller message overhead (as low as 4 bytes), which adapts to network traffic changes to maintain a desired level of security.

Adaptive protocols Prasad et al. [28] suggested using three modes of security: low-level, medium-level, and high-level security; the different levels of security use different cipher algorithms. Depending on the user location (e.g., home vs. public place) or which devices the user is contacting (e.g., trusted vs. untrusted), the model will choose an appropriate security level. The different levels of security achieve different sets of properties, and they use different cipher algorithms. However, choosing the right level of security is not easy; for example, a familiar location (e.g., home) does not necessarily imply the absence of an adversary. Portilla et al. [27] propose changing ECC parameters to provide different levels of security depending on the energy budget of the node. In real world SSL cipher suite provides one form of adaptive security; communicating parties negotiate the cryptographic algorithms to be used at the start of a session. All these three approaches described adapt cryptographic primitives (encryption or MAC algorithms) rather than reducing network overhead, which (as in HnS) would provide more energy savings than the proposed computational adaptation.

The hybrid security model proposed by Shon et al. [31] is the closest work to our adaptive security model. In their adaptive scheme they propose using MAC of different sizes to provide different levels of security, which they choose according to the network characteristics (public, commercial, or private) and the data characteristics (application data or control data). Classifying data sensitivity and determining which level of security would be reasonable for a given data type, while reducing energy used, can be tricky. For mHealth sensing, where the medical data is considered sensitive, their approach would always choose the highest-level security, which would be energy inefficient. Our adaptive model, however, adapts the security level dynamically in response to an attack by an adversary, and in absence of an adversary it adapts to reduce the overhead.

8 Conclusion

We describe a framework for our Hide-n-Sense (HnS) system, which aims to provide a secure and private mHealth sensing environment. We propose the HnS protocol to provide a secure, private, and energy-efficient communication channel between devices in a body-area network. To achieve the desired security and privacy goals and support low-energy sensor devices, we use three techniques: MAC striping, adaptive security, and an asymmetric resource requirement. We demonstrated these techniques using SlyFi, but these techniques can also be applied to other protocols to make them energy-efficient while maintaining their security and privacy properties. Through experiments, we demonstrated that it is feasible to implement and use HnS on low-power devices. In fact, as shown in our experiments, HnS is more energy-efficient than the existing security protocols for low-power sensors, and much more energy-efficient than existing privacy-preserving wireless protocols.

Acknowledgements

This paper results from a research program at the Institute for Security, Technology, and Society at Dartmouth College, supported by the National Science Foundation under Grant Award Number 0910842 and by the Department of Health and Human Services (SHARP program) under award number 90TR0003-01. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

References

[1] S. Agarwal and C. T. Lau. Remote health monitoring using mobile phones and web services. *Telemedicine and e-Health*, 16(5):603–607, June 2010. DOI 10.1089/tmj.2009.0165.

[2] A. Arcelus, R. Goubran, H. Sveistrup, M. Bilodeau, and F. Knoefel. Context-aware smart home monitoring through pressure measurement sequences. In *IEEE International Workshop on Medical Measurement and Applications (MEMEA)*, pages 32–37, Apr. 2010. DOI 10.1109/MEMEA.2010.5480223.

[3] F. Armknecht, J. Girao, A. Matos, and R. L. Aguiar. Who said that? privacy at link layer. In *IEEE International Conference on Computer Communications (INFOCOM)*, pages 2521–2525, May 2007. DOI 10.1109/INFCOM.2007.313.

[4] S. Avancha, A. Baxi, and D. Kotz. Privacy in mobile technology for personal healthcare. *ACM Computing Surveys*, July 2011. Accepted for publication, to appear in 2013, Online at http://www.cs.dartmouth.edu/~dfk/papers/avancha-survey.pdf.

[5] A. R. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003. DOI 10.1109/MPRV.2003.1186725.

[6] D. Bichler, G. Stromberg, M. Huemer, and M. Löw. Key generation based on acceleration data of shaking processes. *Proceedings of Ubiquitous Computing (UbiComp)*, 4717:304–317, 2007. DOI 10.1007/978-3-540-74853-3_18.

[7] F. Buttussi and L. Chittaro. Smarter phones for healthier lifestyles: An adaptive fitness game. *IEEE Pervasive Computing*, 9(4):51–57, Oct. 2010. DOI 10.1109/MPRV.2010.52.

[8] H.-L. Chang, M. J. Shaw, F. Lai, W.-J. Ko, Y.-L. Ho, H.-S. Chen, and C.-C. Shu. U-health: an example of a high-quality individualized healthcare service. *Personalized Medicine*, 7(6):677–687, Nov. 2010. DOI 10.2217/pme.10.64.

[9] TI eZ430 Chronos. http://processors.wiki.ti.com/index.php/EZ430-Chronos.

[10] C. Cornelius and D. Kotz. Recognizing whether sensors are on the same body. In *Proceedings of the International Conference on Pervasive Computing*, Lecture Notes in Computer Science, pages 332–349. Springer, June 2011. DOI 10.1007/978-3-642-21726-5_21.

[11] S. Coyle, F. Benito-Lopez, R. Byrne, and D. Diamond. On-body chemical sensors for monitoring sweat. *Wearable and Autonomous Biomedical Devices and Systems for Smart Environment*, 75:177–193, 2010. DOI 10.1007/978-3-642-15687-8_9.

[12] J.-E. Ekberg and M. Kylänää. Mobile trusted module (MTM):an introduction, 2007. http://research.nokia.com/files/tr/NRC-TR-2007-015.pdf.

[13] Fitbit.com. Fitbit. Online at http://www.fitbit.com/, visited 2010.

[14] O. Garcia-Morchon, T. Falck, T. Heer, and K. Wehrle. Security for pervasive medical sensor networks. In *Proceedings of the International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous)*. IEEE Press, July 2009. DOI 10.4108/ICST.MOBIQUITOUS2009.6832.

[15] S. Gollakota, N. Ahmed, N. Zeldovich, and D. Katabi. Secure in-band wireless pairing. In *Proceedings of the USENIX conference on Security (USENIX Security)*, 2011. Online at https://db.usenix.org/events/sec11/tech/full_papers/Gollakota.pdf.

[16] Green Hills. http://www.ghs.com/.

[17] B. Greenstein, D. McCoy, J. Pang, T. Kohno, S. Seshan, and D. Wetherall. Improving wireless privacy with an identifier-free link layer protocol. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 40–53. ACM, June 2008. DOI 10.1145/1378600.1378607.

[18] L. Huang, K. Matsuura, H. Yamane, and K. Sezaki. Enhancing wireless location privacy using silent period. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, volume 2, pages 1187–1192, Mar. 2005. DOI 10.1109/WCNC.2005.1424677.

[19] D. Kotz. A threat taxonomy for mHealth privacy. In *Proceedings of the Workshop on Networked Healthcare Technology (NetHealth)*. IEEE Press, Jan. 2011. DOI 10.1109/COMSNETS.2011.5716518.

[20] A. Kumar, N. Saxena, G. Tsudik, and E. Uzun. A comparative study of secure device pairing methods. *Pervasive and Mobile Computing*, 5(6):734–749, 2009. DOI 10.1016/j.pmcj.2009.07.008.

[21] R. Mayrhofer and H. Gellersen. Shake well before use: authentication based on accelerometer data. In *Proceedings of the International Conference on Pervasive Computing (PERVASIVE)*, pages 144–161. Springer-Verlag, 2007. DOI 10.1007/978-3-540-72037-9_9.

[22] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: using camera phones for human-verifiable authentication. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 110–124. IEEE Computer Society, May 2005. DOI 10.1109/SP.2005.19.

[23] Microsoft. The HealthVault web-based PHR. Online at http://www.healthvault.com, visited Nov. 2008.

[24] Monsoon power monitor. http://www.msoon.com/LabEquipment/PowerMonitor/.

[25] N. Patwari and S. K. Kasera. Robust location distinction using temporal link signatures. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 111–122. ACM, 2007. DOI 10.1145/1287853.1287867.

[26] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. SPINS: security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, Sept. 2002. DOI 10.1023/A:1016598314198.

[27] J. Portilla, A. Otero, E. de la Torre, T. Riesgo, O. Stecklina, S. Peter, and P. Langendörfer. Adaptable security in wireless sensor networks by using reconfigurable ECC hardware coprocessors. *International Journal of Distributed Sensor Networks*, 2010. DOI 10.1155/2010/740823.

[28] N. R. Prasad and M. Alam. Security framework for wireless sensor networks. *Wireless Personal Communications*, 37:455–469, 2006. DOI 10.1007/s11277-006-9044-7.

[29] L. A. Saxon, D. L. Hayes, F. R. Gilliam, P. A. Heidenreich, J. Day, M. Seth, T. E. Meyer, P. W. Jones, and J. P. Boehmer. Long-term outcome after ICD and CRT implantation and influence of remote device follow-up: The ALTITUDE survival study. *Circulation*, 122(23):2359–2367, Dec. 2010. DOI 10.1161/CIRCULATIONAHA.110.960633.

[30] V. Shnayder, B.-r. Chen, K. Lorincz, T. R. F. Fulford-Jones, and M. Welsh. Sensor networks for medical care. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (SenSys)*, page 314. ACM, 2005. DOI 10.1145/1098918.1098979.

[31] T. Shon, B. Koo, H. Choi, and Y. Park. Security architecture for IEEE 802.15.4-based wireless sensor network. In *Proceedings of the International Symposium on Wireless Pervasive Computing (ISWPC)*, pages 1–5, Feb. 2009. DOI 10.1109/ISWPC.2009.4800607.

[32] D. Singelée and B. Preneel. Location privacy in wireless personal area networks. In *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, pages 11–18. ACM, 2006. DOI 10.1145/1161289.1161292.

[33] J. Sorber, M. Shin, R. Peterson, and D. F. Kotz. Poster: practical trusted computing for mhealth sensing. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 405–406, 2011. DOI 10.1145/1999995.2000058.

[34] J. Sriram, M. Shin, T. Choudhury, and D. Kotz. Activity-aware ECG-based patient authentication for remote health monitoring. In *Proceedings of the International Conference on Multimodal Interfaces and Workshop on Machine Learning for Multi-modal Interaction (ICMI-MLMI)*, pages 297–304. ACM Press, Nov. 2009. DOI 10.1145/1647314.1647378.

[35] D. Walters, A. Sarela, A. Fairfull, K. Neighbour, C. Cowen, B. Stephens, T. Sellwood, B. Sellwood, M. Steer, M. Aust, R. Francis, C. K. Lee, S. Hoffman, G. Brealey, and M. Karunanithi. A mobile phone-based care model for outpatient cardiac rehabilitation: the care assessment platform (cap). *BMC Cardiovascular Disorders*, 10(1):5+, Jan. 2010. DOI 10.1186/1471-2261-10-5.

[36] F. Wartena, J. Muskens, and L. Schmitt. Continua: The impact of a personal telehealth ecosystem. In *Proceedings of the International Conference on eHealth, Telemedicine, and Social Medicine*, pages 13–18. IEEE press, Feb. 2009. DOI 10.1109/eTELEMED.2009.8.

[37] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson. Uncovering spoken phrases in encrypted voice over IP conversations. *ACM Transactions on Information and System Security (TISSEC)*, 13(4):35:1–35:30, Dec. 2010. DOI 10.1145/1880022.1880029.