Adapt-lite: Privacy-aware, Secure, and Efficient mHealth Sensing

Shrirang Mare
Computer Science
Dartmouth College, USA
shrirang@cs.dartmouth.edu

Jacob Sorber
Computer Science; ISTS
Dartmouth College, USA
jacob.m.sorber@dartmouth.edu

Minho Shin
Computer Engineering
Myongji University
South Korea
shinminho@gmail.com

Cory Cornelius
Computer Science
Dartmouth College, USA
cory.t.cornelius@dartmouth.edu

David Kotz
Computer Science; ISTS
Dartmouth College, USA
kotz@cs.dartmouth.edu

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. Although some work on mHealth sensing has addressed security, achieving strong security and privacy for low-power sensors remains a challenge.

We make three contributions. First, we propose Adapt-lite, a set of two techniques that can be applied to existing wireless protocols to make them energy efficient without compromising their security or privacy properties. The techniques are: adaptive security, which dynamically modifies packet overhead; and MAC striping, which makes forgery difficult even for small-sized MACs. Second, we apply these techniques to an existing wireless protocol, and demonstrate a prototype on a Chronos wrist device. Third, we provide security, privacy, and energy analysis of our techniques.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Security

Keywords

mhealth, healthcare, sensor network, network protocol, privacy, security

WPES'11, October 17, 2011, Chicago, Illinois, USA.

1. INTRODUCTION

Recent improvements in mobile computing and developments in miniature medical sensors have enabled mobile health (mHealth) sensing. An mHealth sensing system can deliver continuous health monitoring to patients throughout their daily activities with the potential to simultaneously reduce cost and improve the quality of healthcare [9, for example]. However, there are strong privacy and security concerns associated with mHealth sensing. For instance, a woman wearing a wireless fetal heart monitor, during her daily activities, may not want people around her knowing that she has such a device or observing the data transmitted by the device. In a wireless network, an adversary can observe a device's transmissions and may learn the contents of the transmissions or the type of the device [8, 12]. Thus, mHealth sensing requires protocols that provide strong privacy and security guarantees with low-energy consumption.

There exist some privacy-preserving wireless protocols that provide security and privacy properties like unlinkability, data confidentiality, data integrity, and data authentication. These protocols, however, add various forms of overhead that makes them unsuitable for low-power mHealth sensing networks, which have small sized payloads. Reducing the transmission overhead will make these protocols energy efficient, but reducing the transmission overhead will also makes them less secure (e.g., reducing the message authentication code size will make the protocol less secure against forgery attacks). In this paper we propose techniques to make the privacy-preserving wireless protocols energy efficient by reducing the transmission overhead without significant loss in their security or privacy.

We make three contributions:

1. We propose *Adapt-lite* , a set of two techniques that can be applied to existing wireless protocols to make them energy-efficient (by reducing the transmission overhead) without significant loss in their security or privacy properties. Thus, one can use existing protocols that are otherwise unsuitable for low-power sensor networks due to their large transmission overheads. The techniques are: *i*) *adaptive security*, which dynamically changes the transmission overhead to maintain security while limiting overhead, *ii*) *MAC striping*, which makes a protocol strongly resistant against se-

lective forgery for small-sized message authentication codes (MACs). Adapt-lite provides strong security and privacy when needed (e.g., when a network is under attack), otherwise it uses small transmission overhead to save energy.

2. We applied the Adapt-lite techniques to a variant of SlyFi [2] (a state-of-the-art privacy-preserving wireless protocol), and implemented the modified protocol on TI's eZ430 Chronos wireless devices. We demonstrate experimentally that it is feasible to get strong security and privacy guarantees on low-powered devices. Due to space constraints, we only explain the two Adapt-lite techniques mentioned above. For details on the modified protocol that we implemented, we point the reader to our technical report [3].

3. We provide a security, privacy, and energy analysis of Adapt-lite techniques.

We briefly describe the privacy-preserving wireless protocols in Section 2. We present MAC striping in Section 3, and adaptive-security model in Section 4. Section 5 analyzes Adapt-lite, and we present an evaluation of Adapt-lite in Section 6. In Section 7 we describe the related work.

2. PRIVACY-PRESERVING WIRELESS PROTOCOLS

Due to the broadcast nature of wireless networks, there are many ways in which a user's private information is leaked; for example, an adversary can infer some information about the data being transmitted, or the device transmitting the data, based on any identifying information in the packet (e.g., source or destination node address), the size of the packet, or the timing of the packet sequence. Thus, the goal of privacy-preserving wireless protocols is to obfuscate such information so that the adversary cannot infer any sensitive information about the data or about the device transmitting the data. There are several privacy-preserving wireless protocols proposed in the literature [2, 11]. These protocols provide several properties: *data confidentiality* (an adversary should not be able to learn the contents of the underlying data in the packet), *data authenticity* (an adversary should not be able to forge a packet without being detected), *data integrity* (an adversary should not be able to modify a packet without being detected), and *unlinkability* (given two packets transmitted to/from the same node, the adversary should not be able to link the packets with high probability).

Typically in privacy-preserving wireless protocols, a packet consists of three parts: header, payload, and message authentication code (MAC). To achieve data confidentiality and authenticity the privacy-preserving wireless protocols use standard encryption techniques and message authentication codes respectively, and to achieve unlinkability, these protocols ensure that the header, the payload, and the MAC change with each packet such that the entire packet appears as a pseudorandom string to an adversary. The receiver, however, is able to filter incoming packets using the packet header. We assume that the privacy-preserving wireless protocols use strong cryptographic algorithms to compute the MAC, and choose header non-deterministically such that the only way for the adversary to forge the header or the MAC is by a brute-force attack.

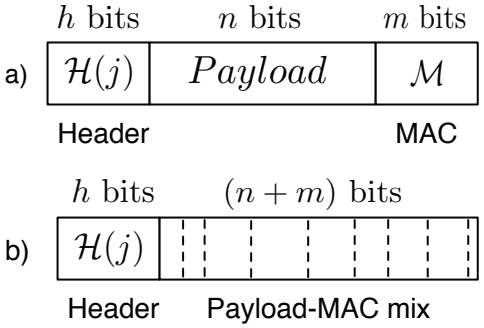

Figure 1: Session packet format: a) Before MAC striping, b) After MAC striping (the dashed lines represent the m MAC bits)

The packet overhead (header and MAC) of these protocols is usually high (e.g., SlyFi [2] overhead is 32 bytes). This overhead is not significant in a Wi-Fi setting, where payloads are large (e.g., 1000 bytes), but for mHealth sensing, where the sensor data is small (1-50 bytes), 32 bytes of overhead is significant.

In this paper we describe the Adapt-lite techniques using the SlyFi protocol, but note that these techniques can be applied to other protocols as well (we discuss this in more detail in Section 6.3).

3. MAC STRIPING

Figure 1a) shows the format of a session packet in the SlyFi protocol. The header $\mathcal{H}(j)$ is a function of the session packet number j. The payload is encrypted using a key (shared during discovery[1]; that is, when the two nodes find each other and share keys to secure the communication), and the MAC $\mathcal{M}$ is computed over the entire packet (i.e., header||payload) using another key (also shared during discovery). Thus, for an adversary to forge a packet with a payload of his choice (i.e., selective forgery), he requires work proportional to 2^{h+m} (he needs to guess the correct h-bit header and m-bit MAC). An adversary, however, may learn the header by intercepting and then disrupting a packet, reducing the selective forgery work to 2^m.

MAC striping changes the packet format slightly (as shown in Figure 1b). The MAC bits are interspersed in the payload at different offsets, represented by dashed lines in Figure 1b. These bit locations are different for each packet, and are generated by a pseudorandom sequence generator function f (we use AES):

$$\langle x_0, x_1, ..., x_m \rangle = f(k, j, n, m), \quad (1)$$

where x_i $(< n+m)$ is the offset for the i^{th} most significant bit of the MAC $\mathcal{M}$, k is the key that was also 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 receiver and sender independently.

When a node (sender or receiver) receives a session message 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 packet of the format in Fig-

[1]Our technical report [3] contains details about the discovery process.

ure 1a). Then the node proceeds with the MAC verification to determine whether the message is authentic.

The advantage of MAC striping is that even with small sized MAC (i.e., small m) it provides strong resistant against selective forgery. We explain this in detail in section 5.

4. ADAPTIVE SECURITY

In wireless protocols the receiver node uses the header to filter incoming packets (that is, to determine whether the incoming packet is addressed to this node), and it uses the MAC $\mathcal{M}$ to verify whether the received packet 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 packet. That is, instead of using a fixed large overhead, the nodes use a small packet overhead that expands dynamically if the node detects the presence of an adversary who is trying to forge a packet.

4.1 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 packet 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 2 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 $\sigma_{SN} \subseteq \sigma_{MN}$.

As shown in Figure 2, 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

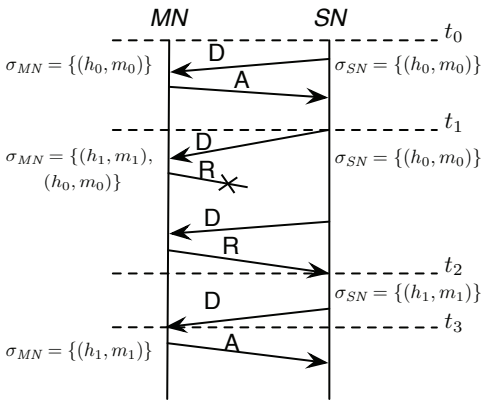

Figure 2: Adaptive security at work (D = data, A = ack, R = reissue)

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 (h_0, m_0)), it receives the message (because $(h_0, m_0) \in \sigma_{MN}$), and replies back to the SN with a reissue command message that tells the SN to start using (h_1, m_1). The SN, however, 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.

4.2 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, the MN does MAC verification to determine whether the message is authentic. If the MAC verification fails, it implies that 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 latter two 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: the probability that an adversary will successfully forge a message should always be less than 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:

$$\beta \geq 1 - (1 - \rho)^a \tag{2}$$

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

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

The MN tracks the number of failed MAC verifications, and conservatively considers each failed MAC verification as a forgery attempt by an adversary. Thus, the probability of a successful forgery when the adversary attempts once is 2^{-m}, where m is the MAC size, and the probability of successful forgery in x forgery attempts is

$$P = 1 - \left(1 - 2^{-m}\right)^x \tag{3}$$

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

$$x \leq \frac{\log(1-\rho)}{\log(1-2^{-m})} \tag{4}$$

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 packet. However, against a clever adversary that intercepts messages to learn valid headers from packets sent by the SN, increasing the header size will not help. 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 4), 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 2 throughout the lifespan of the sensor.

5. SECURITY AND PRIVACY ANALYSIS

SlyFi provides three security and privacy properties: unlinkability, data confidentiality, and data authentication. We explain how Adapt-lite preserves these properties.

Unlinkability.

SlyFi prevents the adversary from linking packets by making each packet seem random to the adversary. It does so by encrypting the entire packet (i.e., header and payload) using AES. Thus, the header, the payload, and the MAC of each session packet is different, and it appears random to an adversary. Adapt-lite truncates the header and the MAC. Since the original header and MAC were pseudo-random strings generated using AES encryption, their truncated strings will also be pseudo-random strings. Therefore, the adversary cannot link session messages to each other in Adapt-lite. Thus, Adapt-lite preserves unlinkability.

Data confidentiality.

SlyFi achieves data confidentiality by using AES (a standard stream cipher) to encrypt the payload. To avoid the adversary identifying two packets having the same payload, it generates different ciphertexts even for the same payload by adding randomness to the encryption: the sequence number changes for each packet, and is used in encrypting both, the header and the payload. Adapt-lite does not modify payload. It truncates the header and the MAC, which are not related to the payload. Thus, Adapt-lite preserves the data confidentiality of the payload.

Data integrity and authenticity.

SlyFi achieves data integrity and authentication by using a message authentication code (MAC). The adversary may attempt to alter the packet content or construct a new packet without being detected; such an attack is called *forgery attack*. We consider two types of forgery attacks: *selective forgery* (when the adversary tries to forge a packet with a payload of his choice), and *existential forgery* (when the adversary tries to forge a packet with any payload).

Selective forgery: In selective forgery the adversary chooses a payload, which is a ciphertext, 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 packet the adversary needs to get the right header and right MAC for a given payload. We, however, assume that the adversary can obtain the right header (by intercepting packets). So for a successful selective forgery the adversary only needs to guess the right MAC for a payload of his choice.

Without MAC striping (Figure 1a), the probability that the attacker succeeds with one random guess is 2^{-m}. With MAC striping, the MAC bits are interspersed with the payload bits at locations determined by Equation 1. Without knowing the key k and the message number j, it is computationally hard for the adversary to determine which bits among the $(n+m)$ bits are the MAC bits, where n is the size of the payload. 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 a $(n+m)$-bits long string (i.e., $payload||MAC$) out of $\{0,1\}^{n+m}$. Out of 2^{n+m} possible such strings, only 2^n payloads exist, thus the success proba-

[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 the knowledge of the encryption key. As a selective forgery attack, however, it suffices to make the MN accept the ciphertext chosen by the adversary.

bility of such a random guess is 2^{-m}. This probability is the same with and without MAC-striping if the guess is made uniformly at random. However, the user can choose the probability of success for a forgery attack, and the adaptive security model will ensure that the probability of a successful forgery is always less than that, as shown in Section 4.2.

Remarks on MAC striping: The adversary cannot easily locate MAC bits in a message because there are $\binom{n+m}{m}$ possible MAC-bit locations and these locations change with every message in a random pattern protected by key k.

6. EVALUATION

To evaluate the efficacy of Adapt-lite, we applied Adapt-lite to a variant of SlyFi and implemented the modified protocol on the eZ430 Chronos wireless wrist device [1] from Texas Instruments (henceforth, for brevity, we will refer to this modified protocol as Adapt-lite). The Chronos wrist device 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. 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 Adapt-lite's ability to respond to an attack, and second on the impact that using Adapt-lite has on the energy consumption of the system.

6.1 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 a protocol at the link layer. The MN and SN use Adapt-lite, 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 valid headers, and only needs to guess the MAC bits in order to forge a packet. 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 3 shows the result of this experiment. The number of security bits used is shown over time for the MN, then SN, and SPINS [5]. SPINS is a security protocol for low-power wireless sensor networks, and we use it as one benchmark for comparison. SPINS uses a 64-bit MAC as protection against message forgery. When not under attack, Adapt-lite achieved much lower overhead than SPINS. During the attack, Adapt-lite increased its overhead; however, the overhead never needed to be raised to SPINS's level, because the attack did not last for very long.

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 $t = 60$ sec the attacker transmits packets with random payload-MAC bits but using a valid header. Such a strong attacker only has to transmit 256 messages to force the MN to increase its MAC size m (see Equation 4).

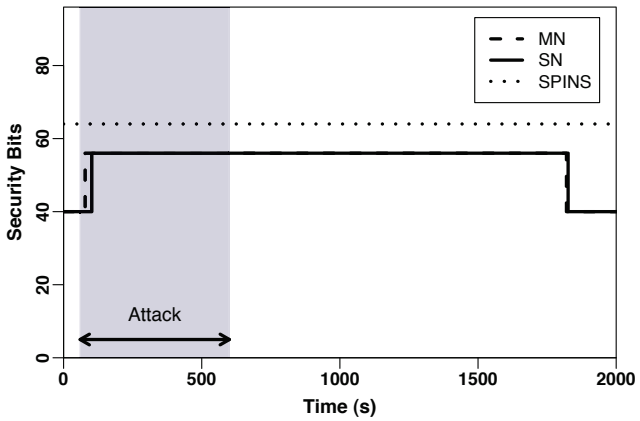

Figure 3: MN and SN adapting in response to an attack.

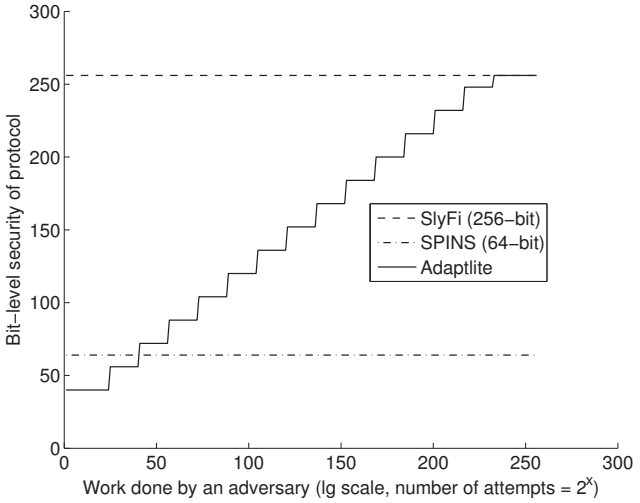

Figure 4: Adaptive security against forgery attack

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 ended, the MN and SN both return to their previous security level after the time period T.

In this experiment, the attack was limited to 10 minutes. Figure 4 shows how Adapt-lite would adapt in the presence of a persistent attack. In this unlikely case, the MN will continue to increase the overhead size (i.e., header+MAC sizes) in response to an increasing number of forgery attempts (log-scale x-axis). The y-axis represents the overhead size (i.e., security bits) used by the protocol. In the plot, initial overhead size is set 48 bits, and ρ is set to 2^{-16}. The plot also compares Adapt-lite's overhead to that of the SlyFi (256-bit) and SPINS (64-bit) protocols.

6.2 Energy analysis

In our second experiment, we measured the energy required to run a SlyFi-like privacy-preserving wireless protocol, SPINS, and Adapt-lite. We simulated the traffic produced by a six-electrode ECG sensor, which sends 10-byte packets (a timestamp and 1 byte per electrode) at a rate of

Table 1: Comparing energy cost of three protocols

	SlyFi	SPINS	Adapt-lite
Overhead	256-bit	64-bit	variable
Security & Privacy[1]	Yes	No	Yes
Avg. 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 Section 5.

50 Hz. We measured the energy consumed using a Monsoon power monitor [4]. Table 1 presents the comparison of the three protocols.

Since communication cost dominated the energy consumption, among the three protocols, Adapt-lite had the least overhead and hence it consumed less power and had greater battery life — twice more than SlyFi and 12% more than SPINS.

6.3 Limitation

Adapt-lite techniques cannot be applied to every protocol. These techniques can be applied to the protocols where only the receiver and sender share the knowledge of a unique value associated with each packet, and this unique value should not be transmitted in the packet. This is true for most privacy-preserving wireless protocols. For example, in SlyFi the header is a function of packet number, and both the sender and receiver can compute it independently, and only the sender and receiver know the packet number; note that the packet number is not the same as the sequential number of the packet as seen on the medium. Thus, Adapt-lite techniques can be applied easily applied to privacy-preserving wireless protocols.

7. RELATED WORK

Adapt-lite presents two techniques: MAC striping, and adaptive-security. There are models related to our adaptive-security model in the literature but to the best of our knowledge, this paper is the first to suggest the MAC striping technique. Compared to our adaptive-security model, the models in the literature differ in how the protocols adapt and under what conditions they adapt.

Prasad et al. [7] suggest using three modes of security: low-level, medium-level, and high-level security; the different levels of security use different cipher algorithms. Portilla et al. [6] propose changing ECC parameters to provide different levels of security depending on the energy budget of the node. Both of these papers adapt cryptographic primitives (encryption or MAC algorithms) rather than reducing network overhead, which (as in Adapt-lite) would provide more energy savings then the proposed computational adaptation.

The hybrid security model proposed by Shon et al. [10] 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 Adapt-lite 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 propose Adapt-lite, a suite of techniques that can be applied to existing privacy-preserving wireless protocols to make them efficient by reducing transmission overhead. Adapt-lite consists of two techniques: MAC striping, and adaptive security. Through experiments, we demonstrate that it is feasible to implement and use Adapt-lite on low-power devices. In fact, as shown in our experiments, using Adapt-lite makes existing privacy-preserving wireless protocols energy-efficient and hence suitable for low-power sensors. We also show that the Adapt-lite techniques preserves the security and privacy properties of these protocols.

9. REFERENCES

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

[2] 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.

[3] S. Mare, M. Shin, J. Sorber, C. Cornelius, and D. Kotz. Hide-n-sense: Privacy-aware secure mHealth sensing. Technical Report TR2011-702, Dartmouth College, 2011. Online at http://www.cs.dartmouth.edu/reports/abstracts/TR2011-702.

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

[5] 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.

[6] 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.

[7] 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.

[8] M. Salajegheh, A. Molina, and K. Fu. Privacy of home telemedicine: Encryption is not enough. *Journal of Medical Devices*, 3(2), Apr. 2009. Online at http://www.cs.umass.edu/~kevinfu/papers/salajegheh-DMD09-abstract.pdf.

[9] 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.

[10] 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.

[11] 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.

[12] 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.