11-3-2006

Securing the Drop-Box Architecture for Assisted Living

Michael J. May
University of Pennsylvania

Wook Shin
University of Illinois

Carl A. Gunter
University of Illinois

Insup Lee
University of Pennsylvania, lee@cis.upenn.edu

Securing the Drop-Box Architecture for Assisted Living*

Michael J. May
University of Pennsylvania
mjmay@seas.upenn.edu

Wook Shin
University of Illinois
Urbana-Champaign
wookshin@uiuc.edu

Carl A. Gunter
University of Illinois
Urbana-Champaign

Insup Lee
University of Pennsylvania
lee@cis.upenn.edu

November 2006

Abstract

Home medical devices enable individuals to monitor some of their own health information without the need for visits by nurses or trips to medical facilities. This enables more continuous information to be provided at lower cost and will lead to better healthcare outcomes. The technology depends on network communication of sensitive health data. Requirements for reliability and ease-of-use provide challenges for securing these communications. In this paper we look at protocols for the *drop-box architecture*, an approach to assisted living that relies on a partially-trusted *Assisted Living Service Provider (ALSP)*. We sketch the requirements and architecture for assisted living based on this architecture and describe its communication protocols. In particular, we give a detailed description of its report and alarm transmission protocols and give an automated proof of correspondence theorems for them. Our formulation shows how to characterize the partial trust vested in the ALSP and use the existing tools to verify this partial trust.

1 Introduction

Advances in networking, distributed computing, and medical devices are combining with changes in the way health care is financed and the growing number of elderly people to produce strong prospects for the widespread use of *assisted living*, a health care approach which can benefit from transferring medical information collected in homes to clinicians over data networks. Software architectures, devices, and protocols for assisted living are all nascent, with expanding deployment of increasingly sophisticated systems. These systems are challenged by strenuous requirements for reliability, usability, and security. So far, these challenges have been addressed at the expense of other aims like the extensibility and openness of the solutions. Closing this gap requires the development of standardized interfaces and protocols whose properties must be assured for all conforming implementations.

One approach to achieving these aims is to employ a division of labor between the Assisted Persons (APs), the clinicians, and the IT specialists where a (possibly independent) monitoring service assures IT-related properties. This parallels the structure of certain other types of home sensor systems such as electronic alarm systems, where a monitoring service like ADT (`adt.com`) acts as an intermediary between home owners and emergency services like police and firemen. The *drop-box architecture* is an approach to assisted living in which a third party known as the *Assisted Living Service Provider (ALSP)* collects medical data from APs and holds it in an encrypted repository from which it can be retrieved by clinicians, the APs themselves, and other parties such as concerned (and authorized) friends and family members. The aim of this paper is to describe a collection of protocols that support the drop-box architecture and analyze their security. Such protocols call for a relatively complex workflow to securely establish a collection of passwords, security tokens, and certificates used by devices from independent vendors, potentially technology-naive APs, diverse general and specialized clinicians, the ALSP, and friends

*In Formal Methods in Software Engineering (FMSE '06), Alexandria, VA, November 2006. ACM.

and family members. These support a suite of protocols that link these parties over a computer network. This system must demonstrably satisfy all regulatory constraints, for example the Health Insurance Portability and Accountability Act (HIPAA) and the Food and Drug Administration's (FDA) rules in the United States.

Once the protocols have been motivated and described, our principal objective is to perform a security analysis of some of their key features. Our work builds on progress toward exploiting the improved formalism of web services to facilitate automated analysis. We describe the drop-box communication protocols as web services, use automated techniques to encode the descriptions, and then employ an underlying theorem proving system to verify security properties. We offer three contributions. First of all are the assisted living protocols themselves. At the current time, assisted living solutions are all proprietary and not available for study beyond high-level information one can obtain from sales brochures and FDA filings. Published protocols could form the basis for standardization leading to open systems that are more capable than unanalyzed single-vendor solutions. Second, we push the boundaries of scalability for formal protocol analysis, which has mainly focused on more generic protocols not so closely tied to a concrete application. This represents a test of the tools which shows their limits and potential for routine usage with more complicated complexes of protocols that involve many types of parties. Third, and finally, we explore a variation on the current tool capabilities by introducing a method to use them to perform a verification on a partially trusted party, *viz.* the ALSP, which collects, stores, and serves medical data but is not privy to the clinical elements of the data itself. Our technique involves compromising the ALSP in a specific way and letting the correspondence theorems imply the desired property.

The rest of this paper is organized as follows. In Section 2 we present the motivation and requirements for assisted living and discuss how these requirements impact security issues for the assisted living protocols. In Section 3 we describe the drop-box architecture and sketch its bootstrapping workflow. Section 4 provides a detailed description of the report and alarm protocols for the drop-box architecture. Section 5 presents our formalization of these protocols, their security model, and describes the verification of theorems for them. Section 6 reviews regulatory issues for assisted living and their relationship to verification and open architectures. Section 7 discusses a prototype implementation for the architecture and protocols. Section 8 describes the related work on assisted living and formal verification of protocols. Section 9 concludes.

2 Motivation

In the United States and many other countries there is a growing number and percentage of elderly people. This population has a greater need for health care services than the younger population. At the same time, in the U.S. at least, health services such as Medicare are moving toward 'episodic' payment systems which emphasize the quality of AP outcomes over a period of time rather than 'per service' payments (such as a payment for each nursing visit or test). This leads to a powerful incentive for care that focuses on lower cost and closer monitoring. When this is combined with steady improvements in networking and medical devices, there is a hope that monitoring can be done on a continuous basis in the homes of APs, thus providing more detailed information and saving the costs and inconvenience of nursing visits to homes or AP visits to a health care facility. A variety of types of such monitoring are now feasible. For example, a pulse oximeter can read pulse and blood oxygen levels with just a simple device that clamps onto a finger. A typical approach is to attach the pulse oximeter to some type of communication hub that connects to the Internet to communicate the results to a clinician. Even as simple a sensor as a bathroom scale can be useful to clinicians since sudden variations in weight are often correlated with significant AP health issues. It is now possible to buy a Bluetooth-enabled scale, which can be associated with a local computer on the Internet that transmits daily weight results to a clinician. More ambitious technologies are on the way. For instance, physicians worry about the APs who fall and become unconscious; devices like accelerometers, which are now provided in devices like motes, could perhaps provide falling alerts. RFID tags may provide very fine-grained information about the locations of APs and objects within the home; this can be used to monitor AP behavior and help APs find things they have misplaced.

There are numerous efforts underway to provide assisted living services based on the motives and technologies just described. However, these services demand much more than a networked medical device. There are essentially three major challenges, as listed in the introduction. The system must be very *reliable* since it is often being used to reduce other types of health care contacts and there is a danger of getting *worse* outcomes rather than better ones. Moreover, the devices are not directly controlled by expert heath

care workers, but instead by APs. The system must therefore be very *usable*. This usability applies to clinicians too: they may be experts with respect to the medical devices and the readings they provide, but may be quite naive about IT issues, such as network reliability, backups, and so on. The system must be *secure* since it deals in sensitive AP data, which is typically sent over a public network. The security issues can be tricky if they involve many parties. Assisted living also faces well-known (but unsolved) challenges for ubiquitous and wireless security. Another factor that impacts reliability and security is the need for regulatory approvals such as those from the FDA and HIPAA in the U.S. Vendors have typically addressed this complex of issues by developing packages in which all or most services are provided by a single coalition of parties. For instance, one approach that is being explored is collaboration between a cable company, device manufacturer, and a health care provider to provide assisted living for an FDA-approved family of devices that talk to a cable set-top box which communicates results to clinicians and allows APs to see their information on their television.

3 Drop-box Architecture

As mentioned earlier, the main drawback to existing approaches is that they are closed and proprietary. This means that a given AP needs to have a service contract with each independent doctor or device that he uses. In particular, existing systems must be extended by the vendor that provides them and cannot be extended by the AP or third party vendors. We have been exploring the development of an open architecture for assisted living called the drop-box architecture. In this section we review its components and bootstrapping protocols.

3.1 Components

The components of the drop-box architecture are illustrated in Figure 1. The architecture is described by Wang, *et al.* [28], including an overview of its potential applications and its implementation using web services. Our aim here is to look in more detail at the security of its communication protocols, so we only sketch the architecture and its rationale briefly here. The main idea is to reduce the IT burden on APs and clinicians by providing an ALSP that operates a server to exchange records. In the drop-box architecture, the medical devices communicate locally with a hub and the hub contacts the ALSP server over the Internet. Records are encrypted at the hub using keys shared only with parties that require them, which means that while the ALSP may see routing information for messages, it is not privy to the clinical contents of the data that it holds. This setup protects all of the parties involved by reducing the number of parties that handle the medical data. In particular, the ALSP can argue that its status is very similar to that of an Internet Service Provider (ISP) and its stored medical records are similar to network audit logs as far as it is concerned. This can all be elegantly implemented using web services, which support formats for encrypting different parts of a message for different parties. Note that the web service approach does not need to rely on tunnels such as IPsec or SSL. Anyway, these tunneling protocols would not support the necessary encryptions by themselves because they do not account for the partially-trusted ALSP intermediary.

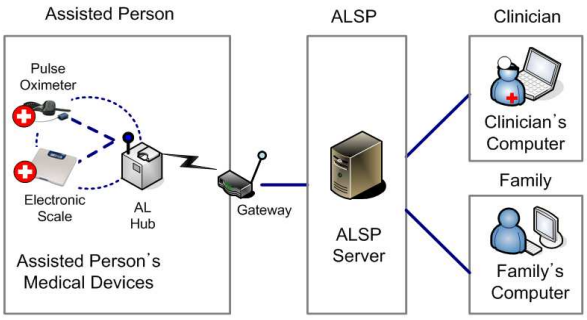

Figure 1: The drop-box architecture.

To sum up the constitution of the drop-box architecture, it consists of the following components: the AP's medical devices, an AL hub (Assisted Living hub), an ALSP server, and computers that clinician and the AP's family and friends use. Medical devices are Bluetooth-enabled light-weight devices which are deployed at the AP's home. Devices send their readings of the AP's physical status to an AL hub over a Bluetooth connection. An AL hub is a dedicated computer that bridges communications between medical devices and an ALSP server. It has the capability to communicate via both Bluetooth and WiFi. Since its job is to secure message transmissions, it must be able to perform cryptographic functions as well. An ALSP server receives and stores the AP's devices readings. The stored readings can be retrieved by authorized users including the AP, clinicians, and authorized friends and family members. Users are authenticated by passwords and entity identification contained in the routing information. The ALSP provides a role based access control system for its database and maintains access control lists for each

AP. The access control system is an essential protection mechanism for the medical records, however, in this work we focus on the security protocols rather than describing the access control system in detail.

3.2 Bootstrapping

Bootstrapping is the initial phase in which an AP, an ALSP, and a clinician build relationships and exchange the following security related parameters: public key certificates (Γ), URL pointers (U^*), user names and passwords (P), DataIDs (δ), access control lists (ACL), and family keys (K_P). In the following discussion we presume that clinicians post their public key certificates on publicly accessible web servers. Clinicians who do not have their own server may post their certificates on a third party certificate authority server or communicate them using some out-of-band technique. Such clinicians may give copies of their certificates to APs via smart cards.

Step 1 The clinician sets up relationships with one or more ALSPs. When the clinician and an ALSP establish their relationship, they exchange URL pointers, U_c^* and U_{AS}^* which respectively point to the public key certificates of the clinician and the ALSP.

Step 2 The AP subscribes to an ISP to get Internet service at home. She purchases a home gateway with wireless connectivity. The gateway manufacturer provides a USB memory stick as an accessory. She uses the memory stick to set up a protected 802.11i (or WPA-PSK) wireless network at home. When she plugs the memory stick into the gateway and presses a "generate key" button on it, the gateway generates a key, K_G and an XML descriptor for the key, $Doc[K_G]$. The gateway copies $Doc[K_G]$ to the memory stick. The memory stick can then be used to add new devices to the home network.

Step 3 The AP visits the clinician who will be monitoring her home medical readings. The clinician asks about her insurance and discusses with her the available medical devices and hubs that her insurance will cover. The clinician then gives her a list of ALSPs that the clinician can use. Finally, the clinician gives the AP a copy of the clinician's public key certificate pointer U_c^*.

Step 4 The AP purchases one or more medical devices and an AL hub. In order to add the AL hub to the home network she plugs the memory stick into the hub's USB port and pushes a "configure" button. The hub then copies the key, K_G from the memory stick and uses it to communicate securely with the gateway from then on. When she turns on a medical device it automatically establishes a Bluetooth connection with the hub.

Step 5 The AP subscribes to an ALSP. The ALSP sends her a URL pointer U_{AS}^* and a password P_{PA} via a USB memory stick. When she plugs the memory stick into the hub, the hub identifies the URL pointer and password on the stick and copies them into persistent memory. When this is done, the AP reports to the clinician that the assisted living network is set up.

Step 6 The AP uses the hub to generate a DataID δ for each medical device. The DataID is used by the ALSP and clinician to identify the AP/device combination that produced a particular reading. Each time a device sets up a connection with the hub, it transmits its serial number s. When the hub detects a new device, the AP can press a button on the hub to generate a new δ for it. The hub then asks the AP which clinicians will receive readings from the device so that their pointers will be associated with it. The hub then associates each clinician pointer U_c^* with the generated δ, storing tuples of the form (s, δ, U_c^*). The hub sends the tuple of (s, δ, AP name) to clinicians so that the clinicians can properly interpret data tagged with δ. The hub also registers the tuple of (δ, U_c^*) so that ALSP can forward data which tagged with δ to authorized clinicians.

Step 7 The AP generates a "family and friends key" K_P. K_P is a symmetric key which encrypts the AP's medical data. Using a smart card reader which is a peripheral of the medical hub, the AP generates several copies of K_P. Several copies of the key are stored in smart cards and shared among the AP's family members or friends. The AP gives the cards to her family members in order to allow them to see her medical data.

4 Report and Alarm Protocols

The drop-box architecture uses a suite of protocols for transmitting information between parties. A complete implementation of the drop-box architecture requires protocols for key distribution and bootstrapping, sending messages from the patient to the doctor, messages from the doctor back to the patient, and a complex role based access control protocol suite to manage the rights of different doctors, patients and family members. We have discussed the bootstrapping protocol above. In this section we focus on a subset of the protocols for sending messages from APs to clinicians.

We have two goals in the transmission protocols: security and privacy.

First, for security our aim is a protocol that is secure against a Dolev-Yao style attacker [15] who can

inject, intercept, or construct arbitrary messages. An attacker cannot, however, guess encryption keys or forge digital signatures. The Dolev-Yao model is limited, however, in that it does not consider the algorithmic problems of combining encryption and digital signatures. Indeed, work by Davis [14], Anderson and Needham [4] and Abadi and Needham [2] have shown that certain combinations of encryption and digital signatures can lead to attacks based on the computational properties of cryptographic primitives. We rely on Davis' proof that the cited attacks can be avoided by using the proper combination of sign-then-encrypt and the inclusion of sender and recipient fields in messages.

Second, for privacy our aim is to prevent an attacker from discovering medical device readings in transit or through straightforward accesses to records stored by the ALSP. To support these requirements we use a combination of authentication and encryption between agents, encrypted storage, and an access control system.

Our trust model assumes that all APs, clinicians, and family members participate honestly in the protocols. Our trust model for the ALSP is as follows. The ALSP's job is to accept messages from APs to store for later access. For security purposes the ALSP is trusted to perform authentication and encryption. However, for privacy purposes, it is not trusted and is not allowed access to the records it stores. An insider attack from the ALSP would therefore break the assumptions about access control, authentication, and availability of records, but would not reveal the contents of the stored medical information. We develop this more formally in §5.2.

Our protocols are based on standard public/private key cryptographic primitives as well as web services security protocols. We adapt the following notation from previous work by the authors in Lug, *et al.* [22]. We assume knowledge of public key certificates. We write Γ for the public certificate, pub(Γ) for associated public key, and priv(Γ) for the associated private key. We write $M\ s : (\text{priv}(\Gamma))$ for a signature on the digest of M using priv(Γ). For clarity, we use two interchangeable notations for symmetric encryption – M_k and M enc:(k). We use for former for encryption of message parts and the latter for whole message encryption.

We use the following notation for salted password authentication and encryption [23].

Definition: (Salted Password Authentication)

We write $A \rightarrow B : M \quad (\text{pswd } P, r, t)$ if A sends B a message of the following form $A \mid M \mid r \mid t \mid \text{MAC}(P, A \mid M \mid r \mid t)$

Here t is the current time according to the clock of A and r is a random number selected by A. The principal B processes this message by checking the following conditions in this order: the time t is not older than a given threshold; the nonce r is not in the replay cache of B; the MAC is correct for the password associated with A. If any of these fails then the remaining steps are omitted and the message is discarded. If all of the conditions succeed, then r is added to the replay cache with an expiration time determined by a given threshold. In this case the message is said to be *valid*. □

The security tokens used in the outline and detailed description of the protocol are named as follows:

Γ_{Doc}: Public key certificate of a doctor (Doc). The associated public key is pub(Γ_{Doc}). The associated private key is priv(Γ_{Doc}).

Γ_{AS}: The ALSP's public key certificate. The associated public key is pub(Γ_{AS}). The associated private key is priv(Γ_{AS}).

K_P: A secret key shared by the AP and her family members.

U^*, V^*: URL pointers to the ALSP's public key certificate.

P_{PA}, P_{FA}: APs and family members passwords (respectively) shared with the ALSP.

ACL_{Pat}: An access control list for the AP's (Pat) records at ALSP.

Report, Alarm: Allowed values for the message-type flag. If the flag is valued Report it indicates the message is non-emergency. If the flag is valued Alarm it indicates the message is an emergency.

4.1 Detailed Protocol: Report

Using the above notation and the names of agents in the system as described in the outline, the protocol steps to transmit a non-emergency *report* are as follows. A schematic of the messages sent appears in Figure 2.

Msg1 $\mathsf{D} \rightarrow \mathsf{H} : n \mid chk(n) \mid t \mid s \mid m$
The AP takes a reading n with her device D. The device computes a checksum of the reading $chk(n)$ and a time stamp t. It concatenates them to its serial number s and sets the message type flag(m = Report) to create Msg1 which it sends to the hub H using Bluetooth link layer encryption.

Msg2 $\mathsf{H} \rightarrow \mathsf{G} \rightarrow \mathsf{AS} : U^*?$
H receives Msg1 from the device and sends its URL reference U^* via the internet gateway G to the ALSP server AS to get the latest version of ALSP's public key certificate.

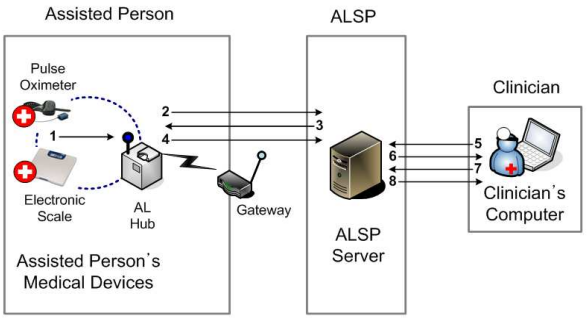

Figure 2: The transmission of a report.

Msg3 $\mathsf{AS} \rightarrow \mathsf{G} \rightarrow \mathsf{H} : \Gamma_{\mathsf{AS}}$
ALSP server's AS receives Msg2, recognizes the pointer for the public key certificate, and sends Γ_{AS} back to the hub via the gateway.

H receives the Msg3 and checks the validity of the certificate provided. If the certificate is valid, it uses $\text{pub}(\Gamma_{\mathsf{AS}})$ to encrypt further messages to AS.

Msg4 $\mathsf{H} \rightarrow \mathsf{G} \rightarrow \mathsf{AS} : \{\{n \mid chk(n) \mid t \mid \delta\}_{K_{mi}} \mid (\mathsf{Family} \mid \{K_{mi}\}_{K_P}) \mid (\mathsf{Doc} \mid \{K_{mi}\}_{\text{pub}(\Gamma_{\mathsf{Doc}})}) \mid \delta \mid m\}$ s:(pswd P_{PA}, r_1, t_1) enc:(K_{mo}) $\mid K_{mo}$ s:(pswd P_{PA}, r_1, t_1) enc:$(\text{pub}(\Gamma_{\mathsf{AS}}))$
H first creates the inner message from the information received from the device, first replacing the serial number s with the DataID δ. It encrypts the whole message with a fresh session key K_{mi}. It then queries the smart card for keys to encrypt K_{mi} under. The smart card returns two keys: K_P for the AP's family and $\text{pub}(\Gamma_{\mathsf{Doc}})$ for the doctor. H then attaches δ and the message type to create the outer message. It signs the outer message with P_{PA} and then encrypts it under a fresh session key K_{mo}. It then signs K_{mo} with P_{PA} and encrypts it under $\text{pub}(\Gamma_{\mathsf{AS}})$.

When ALSP's server receives Msg4, it first decrypts K_{mo} using $\text{priv}(\Gamma_{\mathsf{AS}})$. It then uses K_{mo} to decrypt the outer message. It uses the DataID to identify the message as coming from the AP and then checks the signatures with P_{PA}. If the signatures are valid and the message is not an emergency (m = Report), the reading is queued in the AP's record for retrieval. It checks which recipients are listed and links the encrypted session keys with them, in this case assigning $\{K_{mi}\}_{\text{pub}(\Gamma_{\mathsf{Doc}})}$ with the doctor and $\{K_{mi}\}_{K_P}$ with the AP and her family.

The clinician's office performs a regular checkup of all its APs. It queries each ALSP that the clinician participates with. For each AP, it begins by sending the following message.

Msg5 $\mathsf{CC} \rightarrow \mathsf{AS} : V^*?$
The clinician's computer CC sends V^*, its URL reference, to the ALSP's server AS to retrieve Γ_{AS}.

Msg6 $\mathsf{AS} \rightarrow \mathsf{CC} : \Gamma_{\mathsf{AS}}$
ALSP's server AS checks the URL pointer V^* and sends back Γ_{AS} to the clinician's computer CC.

The clinician's computer CC receives Msg6 and checks Γ_{AS} for validity. If the certificate is valid, it uses Γ to encrypt further messages to AS.

Msg7 CC → AS : {from: Doc | about: Pat | get: new}
s:$(\text{priv}(\Gamma_{\mathsf{Doc}}))$ enc:$(\text{pub}(\Gamma_{\mathsf{AS}}))$
The clinician formulates a request for an AP's records by attaching the doctor's name, the AP's name, and the request type in a message. The message is signed with $\text{priv}(\Gamma_{\mathsf{Doc}})$ and then encrypted with $\text{pub}(\Gamma_{\mathsf{AS}})$. The ALSP has a copy of the clinician's public key certificate on its server and so can authenticate the signature with it.

Msg8 $\mathsf{AS} \rightarrow \mathsf{CC} : \{\{n \mid chk(n) \mid t \mid \delta\}_{K_{mi}} \mid (\mathsf{Doc} \mid \{K_{mi}\}_{\text{pub}(\Gamma_{\mathsf{Doc}})}) \mid \mathsf{Pat} \mid m\}$ s:$(\text{priv}(\Gamma_{\mathsf{AS}}))$ enc: (K_o) $\mid K_o$ s:$(\text{priv}(\Gamma_{\mathsf{AS}}))$ enc: $(\text{pub}(\Gamma_{\mathsf{Doc}}))$
When the ALSP server AS receives Msg7, it first decrypts using $\text{priv}(\Gamma_{\mathsf{AS}})$ and then validates the clinician's signature using Γ_{Doc}. It then checks that the clinician is on the AP's access control list ($\mathsf{Doc} \in ACL_{\mathsf{Pat}}$). The ALSP then sends the new device reading messages along with the AP's name and the encrypted session keys for the clinician. The message is first signed with $\text{priv}(\Gamma_{\mathsf{AS}})$ and then encrypted with a freshly generated message key K_o. K_o is signed with $\text{priv}(\Gamma_{\mathsf{AS}})$ and then encrypted with $\text{pub}(\Gamma_{\mathsf{Doc}})$.

When the clinician's computer CC receives Msg8, it first decrypts the outer message with $\text{priv}(\Gamma_{\mathsf{Doc}})$ and then checks the signature with $\text{pub}(\Gamma_{\mathsf{AS}})$ for validity. The software then decrypts K_{mi} using $\text{priv}(\Gamma_{\mathsf{Doc}})$ and uses it to decrypt the device readings. It checks the checksums for the readings and then enters the new data into the AP's chart.

The AP and her family may retrieve her medical records from the ALSP using a slightly different protocol from the clinician. We use the example of a family member retrieving records. Messages 5 and 6 are omitted because they are similar to above.

Msg7 Family → AS : {from: Family | about: Pat | get: new} s:(pswd P_{FA}, r_2, t_2) enc:$(\text{pub}(\Gamma_{\mathsf{AS}}))$
The requesting family member's computer receives Γ_{AS} in Msg6 and uses it for encryption to the ALSP. It formulates a request for the AP's records by attaching the requestor's name, the AP's name, and the request type in a message. The message is signed with P_{FA} and then encrypted with $\text{pub}(\Gamma_{\mathsf{AS}})$.

Msg8 $\mathsf{AS} \rightarrow \mathsf{Family} : \{\{n \mid chk(n) \mid t \mid \delta\}_{K_{mi}} \mid \{K_{mi}\}_{K_P} \mid \mathsf{Pat} \mid m\}$ s:(priv(Γ_{AS})) enc: (K_o) $\mid K_o$ s:(priv(Γ_{AS})) enc: (P_{FA})

When the ALSP server AS receives Msg7, it first decrypts using priv(Γ_{AS}) and then validates the requestor's signature using P_{FA}. It then checks that the requestor is on the AP's access control list ($\mathsf{Family} \in ACL_{\mathsf{Pat}}$) and notes that the requestor has the role "family" with respect to the AP. The ALSP server then sends the new device reading messages along with the AP's name and the encrypted session keys for family members. The message is signed with priv(Γ_{AS}) and then encrypted with a freshly generated message key K_o. K_o is then also signed with priv(Γ_{AS}) and then encrypted with P_{FA}.

When the requesting family member's computer receives Msg8, it first decrypts the outer message with P_{FA} and then checks the signature with pub(Γ_{AS}) for validity. The software then decrypts K_{mi} using the family member's copy of K_P and uses it to decrypt the device readings. It checks the checksums for the readings and then notes the new data.

4.2 Detailed Protocol: Alarm

In cases of emergency, a variant *alarm* protocol is followed. Emergencies may occur, for example, when a medical device notices that a reading is unusual enough to demand immediate attention. Alternatively, if the ALSP notices that it has not received device readings beyond some safety threshold (*e.g.* 12/24/36 hours) it may trigger an automatic alert. The difference between such an alarm case is that instead of waiting for clinicians to retrieve records, the ALSP actively alerts them. The ALSP must be aware of the relative seriousness of alerts so that it will be able to react appropriately to different situations. For example, an alarm about sudden weight gain from for an AP being watched for anorexia is perhaps not as urgent as an AP being watched for fluid retention. The ALSP does not know the clinical details of the information it receives, so it acts on the basis of an alarm type provided as part of the message. In this case we just assume one type of alarm. When the ALSP server notices that it is facing a potential emergency it takes the following steps.

First, the ALSP inspects the "notify" list for the AP and sends electronic messages to inform them of the situation. The list includes the AP's doctors and may include some authorized friends or family members.

Next, if the ALSP does not receive a quick response from its messages (or if it is after business hours), it attempts to contact the AP by phone to check on her. If it can not reach the AP and establish contact, it phones the clinician's office or messaging service to alert them.

Finally, if all other efforts fail and the ALSP has reason to suspect immediate action is needed, it summons an ambulance to the AP's home for investigation.

We next give a detailed description of how the alarm protocol differs from the report protocol. Since some emergency messaging methods are out-of-band we do not consider them in our detailed description of the protocol. For brevity we only comment on the part that differs from the report protocol above.

Msg1-4 are as above, except that the message type flag is set to alarm ($m = \mathsf{Alarm}$).
When the ALSP's server receives Msg4, it processes it as above. The server notices, however, that the report type is alarm so it takes steps to contact the individuals on the AP's access control list that are marked with a "notify" flag. The AP's doctor is marked with a "notify" flag and so will be sent Msg5 as a notice.

Msg5 $\mathsf{AS} \rightarrow \mathsf{Doc} : \{\text{about:}\mathsf{Pat} \mid \mathsf{Alarm} \mid t_1\}_{\text{pub}(\Gamma_{\mathsf{Doc}})}$ s:(priv(Γ_{AS}))enc:(pub(Γ_{Doc}))

The ALSP server assembles a notify message with the AP's name, a note that there is an alarm, and the previous message's time stamp t_1. The ALSP server AS first signs the message with priv(Γ_{AS}), encrypts it with pub(Γ_{Doc}), and sends it to the clinician Doc.

Since the clinician does not have a server that can accept incoming connections, we denote the message as being sent to her directly rather than to CC. This message will be sent out of band, perhaps to the clinician's email account. The rest of the protocol continues as above with Msg5–8 as given in the report protocol.

5 Protocol Verification

Having specified the report and alarm protocols for the drop-box architecture, we then turned to formalize them and prove some security properties. We specified the protocols formally using the TulaFale [8] specification language. The language has constructs for public key signatures and salted password authentication. The TulaFale script compiles to a script that is verifiable with the ProVerif protocol verifier of Bruno Blanchet (`www.di.ens.fr/~blanchet/crypto-eng.html`,
version 1.13 patch level 6).

5.1 Formal Model

We strove to keep the formal model as close to the protocol specification as possible, even at the cost of some redundancy and additional complexity. This was done for software engineering purposes, to give us confidence that the model truly represents the operation of the implemented system. This effort also ought to be a model for modern push-button protocol verifications–the development of a single high level protocol specification that is directly verifiable using automated tools.

In order to make the model more tractable, however, we made two simplifications. First, we modeled a system with only one AP/doctor pair. We assume that increasing the number of doctors and APs will not break the secrecy proofs, but plan to evaluate this question in future work. Second, we remove the access control functionality from the ALSP, assuming it to be perfect. Since we are verifying only the protocol, rather than the access control system, its removal should not affect the validity of the proofs.

In TulaFale agents are *processes* and messages are passed over named *channels*. Messages are built and taken apart using *predicates*. An attacker can listen to and inject messages on each public channel. The attacker can combine and create messages based on the messages that it hears, but it can not break encryption or forge digital signatures.

To give a taste of the TulaFale script, we now list and explain two sample predicates, the ones needed to build and check (de-construct) Msg7. Note that in a predicate the formal parameters include both the inputs and outputs. Computation is performed by providing some of the parameters as inputs and attempting to derive the rest by executing the statements in the predicate. In our model we name our predicates following TulaFale's convention: mk$M$ to create M and is$M$ to check M.

Example: (Building Msg7)

The predicate to build Msg7 is as follows:

```
1 predicate mkMsg7 (CCcert:bytes,
CCkey:bytes
2  AScert:bytes, CApubkey:bytes,
drname:string,
3  patname:string, Msg7:item,
Msg7signed:item,
4  Msg7encrypted:item):-
5 Msg7 = Message7(drname, patname, "new"),
6 mkSignature(Sig, "rsasha1", CCkey,
7  [<List>Msg7</>]),
8 isX509Cert(AScert, CApubkey, "AS",
"rsasha1",
9  ASpubkey),
10 Msg7signed = <env>Sig Msg7</>,
11 mkAsymEncryptedData(Msg7encrypted,
12  Msg7signed, ASpubkey).
```

Lines 1–4 declare the name of the predicate and the parameter list. Each parameter has a type, `bytes` for numbers and data, `item` for XML tags and pairs, `string` for strings. The meaning of the parameters are: CCcert is Γ_{Doc}; CCkey is priv(Γ_{Doc}); AScert is Γ_{AS}; CApubkey is the public key for the certificate authority; drname is the name of the clinician; patname is the name of the AP; Msg7 is the base message; Msg7signed is it signed; Msg7encryped is it signed and encrypted.

Line 5 creates the Msg7 item as a tuple of the names and request type. Lines 6–7 use the mkSignature library predicate to create a digital signature (Sig) on Msg7 (the brackets are necessary to convert it to a list type) using the specified algorithm (rsasha1) and key priv(Γ_{Doc}). Lines 8–9 use the isX509Cert library predicate to extract the public key pub(Γ_{AS}) from Γ_{AS}. Line 10 creates an "envelope" with Msg7 and its signature (Sig) called Msg7signed. Lines 11–12 use the mkAsymEncryptedData library function to public-key-encrypt Msg7signed with pub(Γ_{AS}). The output is called Msg7encrypted, the message that is sent over the public channel. □

The clinician computer then sends the signed and encrypted message to the monitoring service to request all of the AP's new records. The monitoring service takes it apart using the isMsg7 predicate.

Example: (Checking Msg7)

```
1 predicate isMsg7 (Msg7encrypted:item,
2  Msg7signed:item, CCcert:bytes,
AScert:bytes,
3  ASkey:bytes, CApubkey:bytes,
drname:string,
4  patname:string, Msg7:item) :-
5 isAsymEncryptedData(Msg7encrypted,
6  Msg7signed, ASkey),
7 Msg7signed = <env>Sig Msg7</>,
8 isX509Cert(CCcert, CApubkey, drname,
9  "rsasha1", CCpubkey),
10 isSignature(Sig, "rsasha1", CCpubkey,
11  [<List>Msg7</>]),
12 Msg7 = Message7(drname, patname, "new").
```

As in the previous example, lines 1–4 declare the name of the predicate and the parameter list. The meaning of the parameters are as above with AScert Γ_{AS} replacing CCcert and ASkey priv(Γ_{AS}) replacing CCkey.

Lines 5–6 decrypt Msg7encrypted using ASkey in the isAsymEncryptedData library predicate. Line 7 takes apart Msg7 and its signature. Lines 8–9 take apart Γ_{CC} to extract pub(Γ_{CC}). Lines 10–11 use it to check that Sig is valid for Msg7. Line 12 takes apart Msg7 into its building blocks. □

As part of keeping the model similar to the protocol we introduced some redundancy. First, we wrote two predicates for each message in the report and alarm protocols even though Msg2/3 are similar to Msg5/6 and perhaps could have been merged. We also made use of two public channels, for the gateway and the ALSP, rather than merge them into one. Since the clinician computer does not have a server, in the reporting protocol it correspondingly does not have a named channel to listen on. In the alarm protocol Msg5 must be sent to the clinician, so it is given a third channel to listen on. Note that in the alarm protocol since the clinician computer does not have a copy of the ALSP's certificate until after Msg7, Msg5's verification is deferred until after Msg7's.

While messages between the hub, monitoring service, and clinician service are all secured with message encryption and authentication, messages between the device and hub are secured using link-layer security. The device and hub therefore communicate over a *private* channel that the attacker can not read or write. We do not analyze the security of this communication more deeply. In practice it would be supplied by something like a secured Bluetooth connection.

5.2 Formal results

To test our model we performed three stages of verification on the transmission protocols discussed above. First we examined the correspondence properties of each message and developed the following theorem:

Theorem 1 *(Message Correspondence) The receipt of messages 1, 3, 4, 6–8 as part of the report protocol (1, 3, 4, 5, 7–9 of alarm) corresponds to an authorized process sending them as part of the report (or alarm) protocol.*

Messages 2 and 5 of report (2 and 6 of alarm) are excluded. Both messages are just URL requests for the ALSP's public key certificate. Since the certificate is public knowledge, the ALSP certificate server does not need to authenticate the requestor. The ALSP's response, however, is authenticated so that the recipients know they have received a true copy of the certificate from the ALSP.

Having established correspondences between the messages we examined the receipt of a reading by the clinician. This theorem trusts the monitoring service's verification of messages from APs. If the monitoring service does not perform proper verification, this property is no longer true. It also presumes that the AP is the only source of records at the monitoring service. If others are allowed to create records about the AP (*e.g.* doctors, hospitals, family members) then this property is no longer true.

Theorem 2 *(Readings) Assuming trust in the monitoring service, if a clinician receives a reading* r *with time stamp* t *about an AP from the monitoring service, the specified AP sent it as part of a transmission protocol.*

While we trust the monitoring service for verification of APs and doctors, it is not fully trusted. We do not allow the monitoring service to view the records of the APs that it is holding. To examine this semi-trusted status we sought to prove that the monitoring service can not access readings. Since there is no direct way to ask the prover this, we instead proved a stronger theorem:

Theorem 3 *(Secrecy) If an AP sends a reading* r *encrypted for a doctor or family members, the attacker can not discover it. This is true even if the monitoring service reveals all of its secrets and intermediate information to the attacker.*

To prove this we set up the monitoring service process to publish all of its knowledge to the attacker. A corollary of Theorem 3 is the weaker theorem that we sought to prove:

Corollary 1 *(Secrecy) If an AP sends a reading* r *encrypted for a doctor or family members, the monitoring service can not discover it.*

This corollary validates our semi-trust of the monitoring service by showing that we can separate its role as an authentication and verification agent from its having access to the stored data.

Both protocols involved seven process types (device, hub, gateway, certificate servers). The protocols and queries are encoded in approximately 500 lines of TulaFale code. The queries were performed on an IBM ThinkPad R40 with a 1.4GHz Centrino M processor and 768MB of memory. Running time for the queries ranged from a under 2 minutes with approximately 200MB of memory used for the early message correspondences to 6 minutes with over 360MB of memory for Theorem 2 and several hours with over 600MB of memory for Theorem 3.

6 Regulatory Issues

In the design of our system we must consider the relevant U.S. regulatory issues for health information security and privacy. For our architecture there are two relevant U.S. regulations: the Food, Drug, and Cosmetic Act from the Food and Drug Administration (FDA) and the Health Information Portability and Accountability Act of 1996 (HIPAA) from the Department of Health and Human Services Office for Civil Rights. We discuss shortly how these regulations affect our design, but a complete legal investigation is beyond the expertise of the authors. Our discussion is based on our study of the legal texts and consultation with Food and Drug Adminstration (FDA) Division of Electrical and Software Engineering Deputy Director Brian Fitzgerald.

The HIPAA documents which we consider, the Privacy [18] and Security [17] Rules, apply to only to organizations designated "covered entities", a term which does not include the ALSP. However, since the ALSP will contract with clinicians, it would be a *business associate* of the clinicians [§160.103]. Business associates are beholden to *business associate contracts* which must enforce rules similar to what covered entities must respect [§164.504(e)]. Since the Rules are an upper bound on the restrictions in business associate contracts, for this discussion we consider the responsibilities of the ALSP with respect to the Rules. If the Rules are satisfied, any business associate contract should be as well.

Regarding security, both the Food, Drug, and Cosmetic (FDC) Act [21 USC 301–399] and the HIPAA Security Rule [17] place requirements. A medical device is defined under the FDC Act [21 USC 321(h)] as an "instrument, apparatus, implement, machine, ... which is - (2) intended for use in the diagnosis of disease... or in the cure, mitigation, treatment, or prevention of disease, in man..." which would include parts of the drop-box architecture. This means that the FDA must ensure that it is "safe and effective." Since our architecture proposes the use of existing medical devices in a monitoring architecture, Mr. Fitzgerald remarked that we would need to demonstrate that our architecture works safely and does not cause harm. Since the ALSP stores private health information, it is also faces requirements from the HIPAA Security Rule to ensure data transmission security [§164.312(e)], enforce access control via authentication of users and encryption [§164.312(a)], and implement audit controls and data integrity protections [§164.312(b)–(c)]. While we have only performed a limited security analysis of the protocols, our success at their formalization gives us *provable guarantees* of their efficacy and security against the Dolev-Yao attacker model. One important aspect of the HIPAA Rule is that the yardstick for security is "reasonable and appropriate" security measures. Mr. Fitzgerald remarked that in his opinion our efforts at transmission security would stand up in court by that yardstick. Further work to formalize the bootstrapping workflow, the access control system of the ALSP, and other security related aspects of the drop-box architecture will give us stronger guarantees and thereby greatly ease the way for approval of a deployment of the drop-box architecture.

Regarding privacy, the HIPAA Privacy Rule [18] restricts the disclosures and uses that may be performed on private health information. It requires that holders of private health information such as the ALSP get consent before using or disclosing information under certain circumstances, restricts the transfer of information, and requires holders to provide patients with an accounting of all disclosures of their health information on demand. As with the Security Rule, the Privacy Rule requires "reasonable and appropriate" protections of health information privacy. Our formal proof of the inability of the ALSP to access the information that it holds would give it a defensible claim in court of efforts at privacy protection and shield it from lawsuits by patients accusing it of inappropriate disclosures.

We restrict the ALSP so that it can not access the records that it holds, but even the demographic information about patients and information about the doctors that serve them qualify as protected health information and must be protected [§164.103]. The ALSP will obtain consent from patients when first starting service and adding authorizations for individuals to access their records. Even though this is not required by federal law [§164.506], it is by far common practice and often required by state law. It is possible that patients will need to provide fresh consent for each modification to their access control lists.

It would be reasonable for the ALSP to be on guard for data quality reductions and abrupt cessations of transmissions, either of which may indicate an emergency condition to which it ought to react. We consider such emergency conditions in §4.2.

7 Implementation

After designing and verifying the transmission protocol, we implemented a prototype of the architecture to test its feasibility. Our current prototype demonstrates that it is not difficult to implement the trans-

mission protocol based on the state of the art web service technologies.

In this section, we introduce a drop-box prototype that implements the report protocol which is described in Section 4.1. Using the prototype, an assisted person reports his vital status to a clinician at remote site. When an assisted person tests his vital status using a medical device, the status information is automatically sent to a remote server in a secure manner. The status information is fetched by a clinician's computer periodically so the clinician can trace the physical status of the assisted person. In the remain parts, we explain the hardware devices and the software libraries we used. Next, we describe the message processing procedures and show the message snapshots.

The prototype implementation consists of two notebooks, a desktop machine, and a medical device. The notebooks run Microsoft Windows XP Professional and are connected to the Internet using WPA. The desktop runs Ubuntu Linux operation system and is connected to the Internet using a LAN. The medical device is the Bluetooth-enabled digital pulse oximeter made by Nonin Medical Inc.

The messages are encoded in SOAP Version 1.2 format and transferred via SOAP-RPC. We deployed drop-box services to a Web server using Apache2, Tomcat, v.5.5, and Axis v.1.4. Axis provides WSDL tools and SOAP bindings. The prototype system is written in Java. Consequently, we exploited JCE and Apache XML Security library for Java for securing messages. Additionally, we use AvetanaBluetooth (`http://www.avetana-gmbh.de/`) JSR-82 implementation for our Bluetooth connectivity.

The prototype's message reporting procedure closely follows the description in Section 4.1, but differs in some ways. A gateway is not included in the implementation because it does not contribute towards secure transmissions of the report and the alarm messages. Instead, we just used WPA network of the CS department of UIUC. The asking and retrieving certificates are not included either because the current version of the prototype does not have a certificate server. We assumed that each node already has valid certificates of the other, therefore, Msg2, Msg3, Msg5, Msg6 in the protocol description are omitted. We will upgrade the prototype later to use Java Certificate Service.

The implemented reporting procedure is as follows; The digital oximeter reads assisted person's blood oxygen level and sends it to a notebook that implements the functionality of the assisted person's AL Hub (Msg1). The hub makes a SOAP message, encrypts the message, adds signatures, and sends it

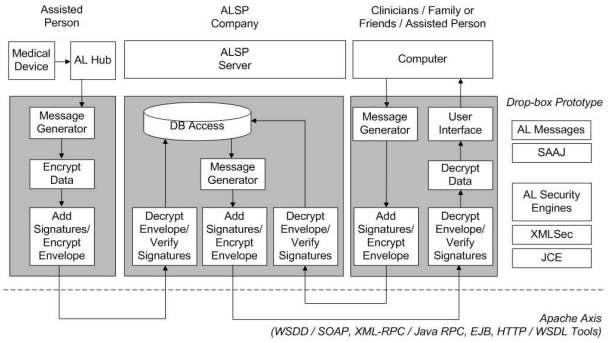

Figure 3: A block diagram of the drop-box prototype system.

to a Linux server (AS) that implements the ALSP's functionality (Msg4). Another notebook (CC), the clinician's computer, sends (AS) a request for new messages (Msg7), and receives the assisted person's physical status information (Msg8). The clinician's notebook is also used as family member's computer. The procedures of a clinician's asking and fetching medical information and those of a family member's are identical except that different cryptographic keys and authentication algorithms are used.

Fig. 3 presents how the above messages are processed in the prototype system. We define AL (Assisted-Living) Messages and generate SOAP envelopes using the SOAP with Attachments API for Java (SAAJ) library. Cryptographic functionalities are supported by AL Security Engines that are built on XML Security library. Our extension from the existing library is to support plural key-encryption-keys. As it is described in the protocol description, our message encryption key is encrypted for doctors and family members separately. We also implemented our own password-based signature functionalities because we could not find proper Java libraries that support XML signature.

Fig. 4 shows a snapshot of Msg1. The message is a plaintext message. It includes a physical status reading of an assisted person, a checksum of the reading, a timestamp, and a device serial number.

H encrypts the message with a symmetric key which is for family members and also with an asymmetric public key of a doctor. It adds the recipient information to the KeyInfo tags. In this implementation, we use Triple DES-EDE and RSA algorithms for the symmetric and asymmetric encryption. Fig. 5 presents the result of the encryption. Fig. 6 shows that signatures to the message and K_{mo} are added to the encrypted message. We implemented a password-based signature functionality using AES so that an

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope ... >
<soapenv:Header>
  <alspns:routeInfo ... />
</soapenv:Header>
<soapenv:Body>
  <alhns:reportBody ... >
    <alhns:patientName ...>Patricia Page</alhns:patientName>
    <alhns:reading ...>/OOT: 0/HR: 67/SpO2: 96</alhns:reading>
    <alhns:checksum ...>...</alhns:checksum>
    <alhns:time ...>Sat Aug 26 12:52:51 CDT 2006</alhns:time>
    <alhns:delta ...>501687</alhns:delta>
  </alhns:reportBody>
</soapenv:Body>
</soapenv:Envelope>
```

Figure 4: A plaintext message which includes physical status information.

```
...
<soapenv:Header>
  <alspns:routeInfo ... />
  <ds:KeyInfo ...>
    <xenc:EncryptedKey ...><xenc:EncryptionMethod .../>
    <xenc:CipherData ...>
      <xenc:CipherValue ...">dPgUTM+cCqhPv...</xenc:CipherValue>
    </xenc:CipherData></xenc:EncryptedKey>
    <alhns:receiver ...>ID_Clinician1</alhns:receiver></ds:KeyInfo>
  <ds:KeyInfo ...>
    <xenc:EncryptedKey ...><xenc:EncryptionMethod .../>
    <xenc:CipherData ...>
      <xenc:CipherValue ...">BW+3sOilGaWSxO1...</xenc:CipherValue>
    </xenc:CipherData></xenc:EncryptedKey>
    <alhns:receiver ...">ID_Family1</alhns:receiver>
  </ds:KeyInfo>
</soapenv:Header>
<soapenv:Body>
  <alhns:reportBody ...>
  <xenc:EncryptedData ...>
    <xenc:EncryptionMethod .../>
    <xenc:CipherData ...>
      <xenc:CipherValue ...>M3hMsTy9vD4KT...</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>
  </alhns:reportBody>
</soapenv:Body>
</soapenv:Envelope>
```

Figure 5: The message after the first encryption.

assisted person and family members can generate signatures using their passwords. Clinicians and the ALSP server use RSA private keys. The final result of the Msg4 is shown in Fig. 7. The message header is encrypted as well as the message body. Only the ALSP can see and verify the signatures.

8 Related Work

The related work for the assisted living system falls into two general categories. First, we consider existing technologies and systems for home health care, both suggested and deployed. Second, we consider work related to the formalization of security and privacy protocols.

There have been many efforts to develop health care systems for people in their own homes or in

```
...
<soapenv:Header>
  <alspns:routeInfo .../>
  <ds:KeyInfo ...>...</ds:KeyInfo>
  <ds:KeyInfo ...>...</ds:KeyInfo>
  <alhns:KeySignatureElement ...>
    <alhns:random>[B@64ea66</alhns:random>
    <alhns:time>...</alhns:time>
    <alhns:signatureValue>[B@79a2e7</alhns:signatureValue>
    <alhns:signer>ID_Patient1</alhns:signer>
  </alhns:KeySignatureElement>
  <alhns:MessageSignatureElement ...>
    <alhns:random>[B@1b60280</alhns:random>
    <alhns:time>...</alhns:time>
    <alhns:signatureValue>[B@14a55f2</alhns:signatureValue>
    <alhns:signer>ID_Patient1</alhns:signer>
  </alhns:MessageSignatureElement>
</soapenv:Header>
<soapenv:Body>
...
```

Figure 6: Signatures are added to the encrypted message.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope ...>
  <xenc:EncryptedData ...>
    <xenc:EncryptionMethod ...>
      <ds:KeyInfo ...>
        <xenc:EncryptedKey ...>
          <xenc:EncryptionMethod ...>
          <xenc:CipherData ...>
            <xenc:CipherValue ...>Z1QRa2ko...</xenc:CipherValue>
          </xenc:CipherData></xenc:EncryptedKey>
        <alhns:receiver ...>ID_ALSP1</alhns:receiver>
      </ds:KeyInfo>
    </xenc:EncryptionMethod>
    <xenc:CipherData ...>
      <xenc:CipherValue ...>P54taGg6+Texs5...</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>
</soapenv:Envelope>
```

Figure 7: The message after the second encryption.

specially-adapted assisted living facilities. Aging-in-Place [12] and Aware Home [25] explore biosensors, audio and video sensors, and RFID tags for monitoring vital signs and movements. In other work, worries about privacy have inspired an emphasis on non-invasive sensors that exclude cameras and microphones [26]. At least one study developed middleware to integrate sensors and medical devices produced by multiple vendors: MiLAN (Middleware Linking Applications and Networks) [24] is a sensor network middleware in which existing network protocols are encapsulated supporting QoS and efficient energy processing. HealthGear [27] from Microsoft Research focused on real-time analysis of physical status data, implemented a client-server architecture using a medical sensor and a cellphone, and validated this with experiments. These studies did not emphasize security considerations. Motiva from Philips [20], Proactive Health from Intel [13], and Healthcare management solutions of IBM [11] proposed home health care architectures, but these all depend on proprietary

products from these respective companies rather than open standards. Moreover, they do not publish detailed workflows and protocol descriptions.

The approach taken to formal methods in this work builds on our work on WSEmail [22], for which we formalized a single protocol and proved a correspondence theorem. Here we have undertaken the formalization and verification of two arguably more complex protocols and proven more complicated theorems. It would be impossible to do justice to the entire body of work on formal methods and protocol verification in this discussion, so we present only selected work that are closely related to our own.

Goodloe, *et al.* [19] analyze a multi-party protocol for network service billing using logical simulations. Their protocol has an intermediary, the Network Access Server (NAS), which is similar to our ALSP in some respects, but they focus on functional properties of the protocol rather than proving a correspondence theorem.

Efforts at the formal verification of messaging protocols include Zhou, *et al.*'s [29] work on verifying the properties of Privacy Enhanced Mail (IETF RFCs 1421–4) and Abadi, *et al.*'s [1] formal proof of correctness for a trusted third party messaging system.

For our formal analysis we used the TulaFale language, a product of the Samoa (`securing.ws`) project at Microsoft Research. The authors of the project have developed theoretical frameworks and for protocol verification in the context of web services [6, 8] and general security protocols [7]. The TulaFale language compiles to the input language for Bruno Blanchet's ProVerif protocol verification tool. The tool verifies protocols using the Dolev-Yao assumption of perfect cryptographic primitives. As noted above, the model ignores the computational aspects of cryptography. Although this makes its proofs incomplete [5], such proofs have a strong record for detecting errors and recent work on an protocol verification from the computational perspective [9, 3] may offer new avenues of automated exploration.

Designers of home health care and telemedicine systems acknowledge the need for security in their design (*e.g.* [16, 21, 10]), however there has been little focus on formal verification. Instead, studies largely show software correctness using UML and use security tunnels based on SSL. Such an approach skirts the issues of protocol correctness and security, since simply using secure tunnels does not guarantee end-to-end message security and privacy. Indeed, as we mentioned earlier, SSL alone cannot implement the drop-box architecture.

9 Conclusion

The drop-box architecture for assisted living is characterized by a semi-trusted Assisted Living Service Provider (ALSP) that houses data from patients for later access by clinicians and others. The architecture is designed to create a reasonable distribution of IT responsibilities and reduce the burden on patients and clinicians in managing transfer of home health care records. The architecture relies on standards-based communication and security protocols and is designed to allow for open interfaces between communicating parties.

In this work we have developed the fundamental workflows and communication protocols needed for its operation. We first describe the protocols for the transmission of both normal data *reports* and emergency data *alarms* in detail and then translate them to the TulaFale typed protocol language.

We use the ProVerif security protocol verifier to derive theorems about message correspondence and end-to-end transmission authentication. As a secrecy property we prove that the ALSP can not access the private data that it holds. We show this property by first showing that even when the ALSP publishes all of its secrets the attacker can not break the secrecy.

This work is a continuation of work by the authors on the design and specification of practical, application-oriented security protocols that are easily subjected to formal verification. The availability of languages for high level specification of protocols and tools for push-button verification their security in the face of attackers promise to ease the task of formal protocol verification. This study shows that existing tools are reaching a good level of feasibility for realistic protocol suites.

Acknowledgements

We would like to thank to Brian Fitzgerald and Paul Jones of the FDA and the UIUC Assisted Living Project for discussions about this work. We also benefited from discussions with Michael Hicks and Michael Nidd. This research was supported in part by NSF CNS 05-06546, NSF CNS 05-09268, ARO W911NF-05-1-0158, NSF CCF 04-29948, and ARO DAAD19-01-1-0473.

References

[1] M. Abadi, N. Glew, B. Horne, and B. Pinkas. Certified email with a light on-line trusted third party: design and implementation. In *Proceedings of the eleventh international conference on World Wide Web*, pages 387–395. ACM Press, 2002.

[2] Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. Research Report 125, Digital Systems Research Center, Palo Alto, CA, 1 June 1994.

[3] Martín Abadi and Philip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.

[4] Ross J. Anderson and Roger M. Needham. Robustness principles for public key protocols. In Don Coppersmith, editor, *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 236–247. Springer, 1995.

[5] Michael Backes and Matthias Schunter. From absence of certain vulnerabilities towards security proofs: pushing the limits of formal verification. In *NSPW '03: Proceedings of the 2003 workshop on New security paradigms*, pages 67–74, New York, NY, USA, 2003. ACM Press.

[6] K. Bhargavan, C. Fournet, and A. Gordon. A semantics for web services authentication. In *Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on principles of programming languages*, pages 198–209. ACM Press, 2004.

[7] K. Bhargavan, C. Fournet, A. Gordon, and S. Tse. Verified interoperable implementations of security protocols. In *19th IEEE Computer Security Foundations Workshop (CSFW-19)*, pages 139–152, Venice, Italy, July 2006. IEEE Computer Society.

[8] K. Bhargavan, C. Fournet, A. D. Gordon, and R. Pucella. TulaFale: A security tool for web services. In *International Symposium on Formal Methods for Components and Objects (FMCO'03)*, LNCS. Springer, 2004.

[9] Bruno Blanchet. A computationally sound mechanized prover for security protocols. In *IEEE Symposium on Security and Privacy*, pages 140–154, Oakland, CA, May 2006. IEEE.

[10] Zhe Chen, Xiaomei Yu, and David Feng. Telemedicine system over the internet. In *CRPITS '00: Selected papers from the Pan-Sydney workshop on Visualisation*, pages 113–118, Darlinghurst, Australia, Australia, 2001. Australian Computer Society, Inc.

[11] IBM Corporation. Healthcare management solutions. `www.research.ibm.com/hc/home.html`.

[12] Intel Corporation. Age-in-place. `www.intel.com/research/prohealth/cs-aging_in_place.htm`.

[13] Intel Corporation. Proactive health. `www.intel.com/research/prohealth/`.

[14] Donald T. Davis. Defective sign and encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML. In *Proceedings of the Usenix Technical Conference*, pages 65–78, Boston, MA, June 2001.

[15] D. Dolev and A.C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1984.

[16] Anamarija Margan Edgar Pek, Sven Loncaric. Internet-based medical teleconsultation system. In *Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis*, pages 657–661, Pula, Croatia, June 2001. IEEE R8-EURASIP.

[17] Office for Civil Rights. Health insurance reform: Security standards; final rule. *Federal Register*, 68(34):8334–8381, 20 February 2003. 45 CFR Parts 160, 162, and 164.

[18] Office for Civil Rights. Standards for privacy of individually identifiable health information. Regulation Text (Unofficial Version) 45 CFR Parts 160 and 164, U.S. Department of Health and Human Services, August 2003. As amended: May 31, 2002, Aug 14, 2002, Feb 20, 2003, and Apr 17, 2003.

[19] A. Goodloe, C. A. Gunter, and M. Stehr. Formal prototyping in early stages of protocol design. In *WITS '05: Proceedings of the 2005 Workshop on Issues in the Theory of Security*, pages 67–80, New York, NY, USA, 2005. ACM Press.

[20] Philips Inc. Motiva interactive healthcare platform. `www.medical.philips.com/main/products/telemonitoring/products/motiva`.

[21] V. Jones, A. Rensink, T. Ruys, E. Brinksma, and A. van Halteren. A formal MDA approach for mobile health systems. In *Second European Workshop on Model Driven Architecture (MDA)*, Canterbury, England, Sept 2004.

[22] Kevin D. Lux, Michael J. May, Nayan L. Bhattad, and Carl A. Gunter. WSEmail: Secure internet messaging based on web services. In *International Conference on Web Services (ICWS)*, Orlando, FL, July 2005. IEEE.

[23] Anthony Nadalin, Phil Griffin, Chris Kaler, Phillip Hallam-Baker, and Ronald Monzillo (Eds.). Web services security UsernameToken profile 1.0. Standard 200401, OASIS, March 2004.

[24] Univ. of Rochester. Center of future health. `www.futurehealth.rochester.edu/news/`.

[25] Georgia Institute of Technology. Awarehome. `www.cc.gatech.edu/fce/ahri/`.

[26] Univ. of Virginia. Smart in-home monitoring system. `marc.med.virginia.edu/projects_smarthomemonitor.html`.

[27] N. Oliver and F. Flores-Mangas. HealthGear: A real-time wearable system for monitoring and analyzing physiological signals. Technical Report MSR-TR-2005-182, `research.microsoft.com/~nuria/healthGear/`, Redmond, WA, 2005 May 2005.

[28] Q. Wang, W. Shin, B. K. Alshebli, M. Caccamo, C. Gunter, E. Gunter, J. Hou, K. Karahalios, X. Liu, C. Oh, L. Sha, and Z. Zeng. An open system architecture for assisted living. In *IEEE International Conference on Systems, Man, and Cybernetics*, Taipei, Taiwan, Oct 2006.

[29] D. Zhou, J. Kuo, S. Older, and S. Chin. Formal development of secure email. In *Proceedings of the 32nd Hawaii International Conference on System Sciences*. IEEE Computer Society, 1999.