SECURITY SOLUTIONS FOR CYBER-PHYSICAL SYSTEMS

by

Krishna Kumar Venkatasubramanian

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

ARIZONA STATE UNIVERSITY

December 2009

SECURITY SOLUTIONS FOR CYBER-PHYSICAL SYSTEMS

by

Krishna Kumar Venkatasubramanian

has been approved

September 2009

Graduate Supervisory Committee:

Sandeep Gupta, Chair
Partha Dasgupta
Rida Bazzi
Dijiang Huang

ACCEPTED BY THE GRADUATE COLLEGE

ABSTRACT

Cyber-Physical Systems (CPS) are sensing, communication and processing platforms, deeply embedded in physical processes and provide real-time monitoring and actuation services. Such systems are becoming increasing common in enabling many of the pervasive computing technologies that are becoming available today such as, smart-homes, smart-vehicles, pervasive health monitoring systems. Given the automation that CPSs introduce in managing physical processes, and the detail of information available to them for carrying out their tasks, securing them is of prime importance.

In this dissertation, a novel security paradigm for CPSs is proposed, called Cyber-Physical Security (CYPSec). CYPSec solutions are unique in that they take they take into account the environmentally-coupled nature of CPSs in enabling security solutions. This dissertation explores CYPSec solutions for two diverse but related problems. The first is a usable and secure key agreement protocol called Physiological Signal based Key Agreement (PSKA), which combines signal processing and cryptographic primitives to enable automated key agreement between sensors in a Body Area Network (BAN) without any form of external user involvement. It uses specific physiological stimuli-based features (Photoplethsymogram and Electrocardiogram) from the human body for its task. The second is an access control model called Criticality Aware Access Control (CAAC), which facilitates a more adaptive and proactive provisioning of authorizations - provide the right set of privileges for the right set of subjects, at the right time for the right duration - for managing emergencies within smart-infrastructures.

The following are the principal contributions of this dissertation: 1) a novel CYPSec solution for BANs (PSKA) - which combines physiological signal processing and cryptographic primitives for securing inter-sensor communication; 2) a benchmark of PSKA using Matlab to demonstrate its correctness and usable security design goal; 3) successful prototype of PSKA on Crossbow Mote platform as a part of *Ayushman* pervasive health monitoring test-bed, to demonstrate its viability on resource-constrained platforms in terms of computation, communication, memory and energy consumption requirements; 4) a CYPSec access control model for smart-infrastructures (CAAC) - which can facilitate dynamic and proactive emergency management by temporarily providing the required privileges to users without their explicit request; 5) a detailed formalization of CAAC, along with description of its policy specifications, and an example usage scenario on a smart-oil rig platform; and; 6) a prototype of CAAC as a part of the Ayushman test-bed to demonstrate its proactivity and adaptiveness design goals.

À mes chers parents.

ACKNOWLEDGMENTS

I sincerely thank my advisor, Dr. Sandeep K. S. Gupta, for his guidance, encouragement and for providing the opportunity to work on novel and challenging research problems. His constant attempts to push me out of my comfort zone, has helped tremendously in improving my research and communication skills. I would also like to thank my dissertation committee members, Dr. Rida Bazzi, Dr. Partha Dasgupta, and Dr. Dijiang Huang, for their advice on my research, and their time and effort to help me fulfill the degree requirements.

I thank my present and past colleagues of the IMPACT Lab Guofeng Deng, Qinghui Tang, Su-Jin Kim, Gianni Giorgetti, Sarma Upadhyayula, Vikram Shankar, and Georgios Varsamopoulos, for making my PhD experience a memorable one. In particular, I thank Tridib Mukherjee for his valuable insights and always being game for a discussion, brainstorming or critiquing session. Our evening “coffee-promenades” featured conversations on diverse topics from politics to philosophy, and were always refreshing and enjoyable. A special thanks to Ayan Banerjee, with whom I had the opportunity to work closely, for his hard-work, Matlab skills, and a general willingness to help out.

I thank my friend and former room-mate of seven and half years, Punarvasu Joshi, for his encouragement, support, and honest opinions during my years of graduate studies.

I thank the National Science Foundation (through grants CNS-0617671 and CT-0831544), MediServe Information Systems, and Consortium for Embedded Systems at Arizona State University for funding this research.

But most importantly, I thank my dear parents whose love, understanding, advise, encouragement, and much more made this journey possible.

Thank you all!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

1. INTRODUCTION

Our society has been facing considerable challenges in recent years. Increasing traffic congestion, energy scarcity, rising medical costs, climate change and many other issues have taken a turn for the worse and need urgent attention. Technology can play a major role in alleviating these problems through the development of smart-infrastructures.

The idea behind smart-infrastructures is to incorporate intelligence in everyday objects/services in order to improve the efficiency of performing certain rudimentary but crucial tasks. For example, a smart coffee pot can detect the decrease in temperature of its contents (coffee) and alert the user so that the coffee does not have to be unnecessarily re-heated; thereby saving energy. This trend of developing intelligent systems has already begun. A recent survey found that a typical household has at least 100 microprocessors while a typical new model car has more than 100 of its own [12]. In fact, most of microprocessors are now embedded in systems which are not computers [65]. The crucial technology that has made this leap possible are miniature sensing, communication and processing platforms which can be embedded as a part of larger systems/processes for providing real-time monitoring and feedback control services [40]. Such platforms, deeply embedded in physical processes, are called *cyber-physical systems* [120].

The principal goal of *Cyber-Physical Systems (CPS)* is to monitor the behavior of physical process[1] they are a part of, and actuate actions to change its behavior, if needed. CPS platforms are usually designed as an amalgamation of electro-mechanical sensors and actuators, a communication stack, memory and a processing unit. Each of these components can be *centralized* - in one entity for example, implantable cardiac defibrillators (ICDs) or pacemakers which are embedded inside a person's chest cavity (as a part of the cardiovascular process) to observe its behavior (cardiac cycle) and correct in the event of out of the ordinary behavior (arrhythmia) [121], or *distributed* over a group of entities as in the case of an automobile control system, where sensors from the engine provide temperature data to a microprocessor dedicated to manage engine functionality, which then communicates the data through an in-car network to a controller in dashboard which displays this information to the driver.

Cyber-Physical systems have applications in many domains such as health management, vehicular man-

[1]Henceforth, we use the term physical process interchangeably with the term environment

agement, data-centers, power-grids, and physical infrastructure (roads, bridges). Irrespective of the domain, a CPS has three principal characteristics:

- *Environment Coupling:* CPSs are very tightly coupled with their environment (physical process) - any change in the behavior of the environment results in a change in the CPS' behavior and vice-versa. Prominent examples include medical devices such as ICDs.
- *Diverse Capabilities:* CPSs are usually made up of diverse heterogeneous entities with order of magnitude difference in capabilities. Sensors deeply embedded in physical processes for monitoring purposes have limited capabilities, while those entities which manage them are much more capable. For example, a health monitoring CPS is usually made up of limited size medical sensors for usability reasons, but the base station managing the sensors is usually a hand-held computer. A direct consequence of this heterogeneity is potential bottleneck in terms of computation, communication and memory in the workflow.
- *Networked:* CPSs, unlike traditional stand-alone embedded systems, usually require a communication channel between its components, either embedded within the physical processes or external to it, in order to provide its (usually coordinated) services [64]. For example, in an automobile CPS, a sensor monitoring the car transmission communicates with the car radio in order to enable it to increase the volume automatically as the speed increases, thus compensating for the extra noise.

Many issues need to be addressed in order to make each of these CPS system types viable such as managing the cyber and physical interactions, ensuring safety, energy-efficiency, interoperability, sustainability. One of the most important aspects that needs to be considered in this early development phase of cyber-physical systems is security. Figure 1 illustrates CPS its need, application, issues and properties.

1.1. Security for Cyber-Physical Systems

Security is of prime importance in Cyber-Physical Systems. In this section we present some of the principal aspects of securing CPS from the need for security in CPS, the requirements of securing a typical CPS, followed by the challenges in meeting the requirements.

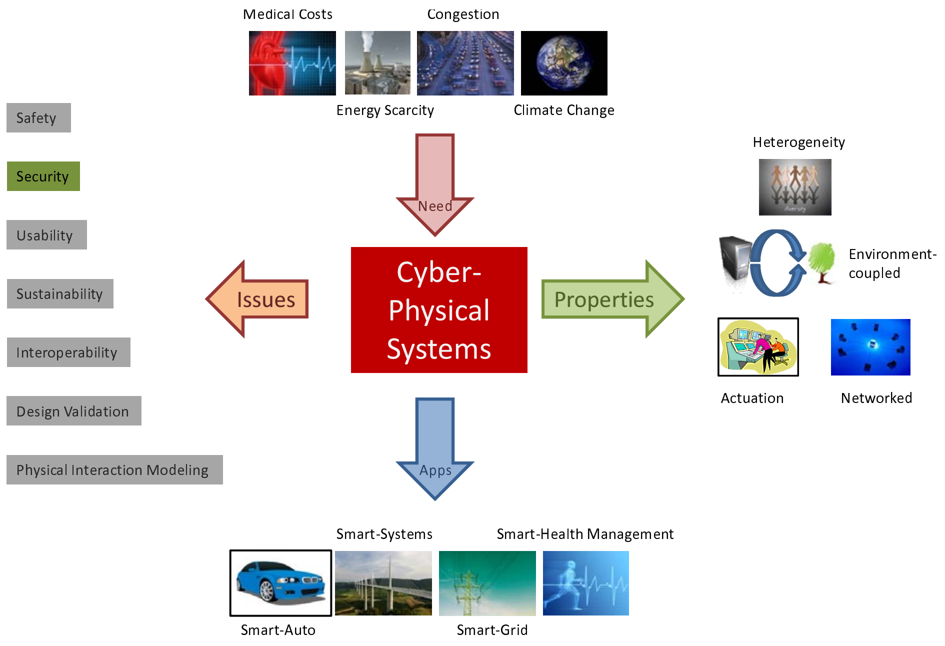


Fig. 1. Cyber-Physical Systems

1.1.1. Need for CPS Security

CPSs, given their environmental coupling, diverse capabilities and lack of isolation are often used for monitoring and controlling *mission critical*, processes. Therefore any security compromise of the CPS can have profound consequences. Further, the mission critical nature also makes them more susceptible to targeted attacks. The case in point is the pace-makers CPSs which have been targeted to not only reveal a patient's electrocardiogram (EKG) data but also to actuate an untimely shock [28]. Further, CPSs have the ability to monitor the physical process they are embedded in. This makes them privy to detailed and often sensitive information about the process. If this information is available to malicious entities, it can be exploited leading to loss of privacy, abuse and discrimination. For example, unauthorized knowledge of the electricity consumption of a neighborhood from a power-management CPS in the wrong hands can result in socket bombing attacks on households perceived to be using excessive electricity. Finally, CPSs have the ability to actuate changes to the environment they are a part of. Allowing unauthorized parties to actuate untimely changes to the environment can cause harm to the process itself. For example, malicious entities can easily shut-down a CPS controlling an automobile leading to issues ranging from inefficient fuel consumption to break-failure. In a world where we are becoming increasingly dependent upon CPSs to provide us with automated, efficient management of essential services, care has to be taken to ensure that they are protected.

However, before we delve into these details of how to alleviate CPS security issues, we need to understand the typical workflow of CPS. This will provide information on the potential security issues and bottlenecks that have to be tackled.

1.1.2. Cyber-Physical System Workflow

We can categorize the workflow of CPSs into three main functions: 1) *Monitoring*: This is the most fundamental aspect of CPS and deals with monitoring the environment in which the CPS is functioning. It is also used to provide feedback on any past actions taken by the CPS and ensure correct operations in the future; 2) *Processing*: This deals with analyzing the data collected during monitoring to determine whether the physical process is meeting certain pre-defined criteria. In situations where the criteria are not being satisfied, the corrective actions are determined which when executed ensures the criteria is satisfied. For example, a data-center CPS can observe its thermal characteristics, develop a model to predict the temperature rise with respect to various scheduling algorithms, which can be used to determine future job [121]; and 3) *Actuation*: This deals with executing the actions determined during data processing phase. Actuation can take many forms from changing the cyber behavior of the CPS, to changing the physical process itself. For example, delivery of medicine in a medical CPS, and authorizing information access in a smart-space CPS.

This CPSs' workflow can execute in one of three possible modes: 1) *Passive* - in this mode, CPS act as information gathering platforms only and solely monitor their environment, process the data, e.g., medical devices such as pulse oximeters; 2) *Passively Active CPS* - in this mode, CPS monitor their environment (physical aspect) and if they see some property not being satisfied they execute an *indirect actuation* by changing their own behavior (the cyber aspect) so that the property can be satisfied, example, data centers performing smart scheduling to minimize temperature rise at specific spots; 3) *Active CPS* - in this mode, CPS monitor their physical environment and if some property is not being satisfied they execute a *direct actuation* by modifying the behavior of the physical environment to ensure the property is satisfied, example,

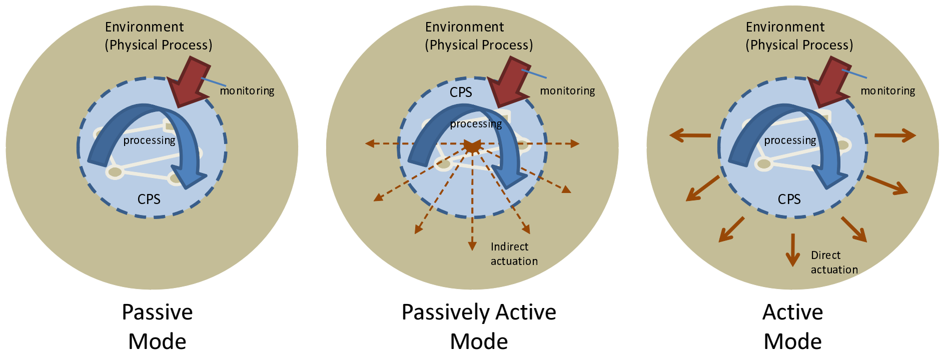


Fig. 2. CPS Workflow and Operating Modes

an HVAC (heating, ventilating, and air conditioning) unit in a building blowing cold air into the building if the temperature goes beyond a certain established limit. Figure 2 illustrates the CPS types. Now that we have an overview of a typical CPS workflow, we can identify and tackle its security requirements.

1.1.3. Security Requirements

Given the recent trend toward complex and open design, use of completely-off-the-shelf (COTS) components and interconnection with existing insecure global communication infrastructure such as the Internet, security for CPSs has become very important. It can be seen that CPSs are expected to perform diverse set of operations not just directed toward modifying the physical process but also to change its own behavior as well. The workflow and the aforementioned characteristics of CPS illustrate many of the principal security requirements of CPSs, which any security architecture for CPSs should be designed to meet:

Sensing Security: As CPSs are closely related to the physical process they are embedded in, the validity and accuracy of the sensing process has to be ensured. Sensing Security needs techniques for *physical stimuli authentication*, so that any data measured from the physical process can be trusted.

Storage Security: Once the data has been collected and processed it may be required to be stored over time for future access. Any tampering of this stored data can lead to errors in future data processing requirements. Storage Security involves developing solutions for securing stored data in CPS platforms from physical or cyber tampering.

Communication Security: An important aspect of CPSs is that they are networked in nature. This not only allows them to form a network for data fusion, and delivery to back-end entities but also take coordinated response actions (in both the passively active and active operational modes). Communication Security needs the development of protocols for securing both inter and intra-CPS communication from both active (interferers) and passive (eavesdroppers) adversaries.

Actuation Control Security: This refers to ensuring that during the passively active or active mode of operation, no actuation can take place without the appropriate authorization. The specification of the authorizations has to be dynamic as CPSs' requirements change over time.

Feedback Security: This refers to ensuring that the control systems in a CPS which provide the necessary feedback for effecting actuation are protected. The current security solutions focus on data security only, but their effects on estimation and control algorithms has to be studied for providing an in-depth defense against CPS [17].

Figure 3 shows the security requirements for a typical CPS. Security for cyber-physical systems is a relatively new area and not much work has been done in this realm. As with any new field most of the effort seems to be focused on mapping solutions from existing domains such as sensor networks which share the networked operation and low capability characteristics with CPS [118]. However, these solutions being not designed for CPSs are not efficient. For example, establishing secure communication channel between two nodes in a sensor network assumes manual pre-deployment of keys at the nodes, as it saves energy during the pivotal key distribution phase. This however is very inefficient for CPS and disrupts their workflow, given their deeply embedded nature [127].

1.1.4. Challenges in Cyber-Physical Security

Securing CPSs present many challenges which need to be considered given their monitoring and actuating nature. Some of the fundamental challenges that need to be addressed include:

- *Altered Threat Model:* The traditional threat model for computational systems has been solely focused on cyber threats. But the environmentally-coupled nature of CPS means that tampering with the physical environment around the CPS may result in failure of CPSs to function correctly. As an extension, attackers need not tamper with the environment itself, but the sensing process which might cause the CPS and its security apparatus to malfunction.

- *Application Specific Security Requirements:* Traditional security solutions are expected to meet requirements such as confidentiality, integrity, and availability for the entire system. These requirements are not static and many more may be needed based on the application. CPSs follow a similar application-specific security requirement property however with two fundamental differences: 1) each function of CPS - sensing, communication, storage, actuation and feedback - has its own set of security requirements, and 2) what it means to enable these requirements is function dependent. For example, actuation authorization in a medical monitoring CPS may mean that delivering a particular volume of drug delivery has been sanctioned, while in a smart-space CPS it might mean facilitating (not executing) a response action for an emergency.

- *User-Centric Security:* The deployment of CPSs is not limited to specialized systems managed by tech-savvy people. Many of the applications of CPSs are systems of every-day use operated by non-technical people - medical monitoring systems, smart-infrastructures and so on. Therefore security solutions for CPSs should have a high degree of *usability* - plug-n-play nature and security transparency - a characteristic that today's cyber-only security solutions do not consider.

1.2. Approach: Cyber-Physical Security Solution

We believe that in designing security solutions for such CPSs, one should not only consider the characteristics of the computational components involved (CPU, RAM, ROM, data rate), but also the interaction of the components with the physical environment. In this work we therefore present a novel perspective on securing CPS which takes this property into account, called **Cyber Physical Security Solutions (CYPSec)**. CYPSec solutions are *environmentally-coupled security solutions*, which take traditional security primitives

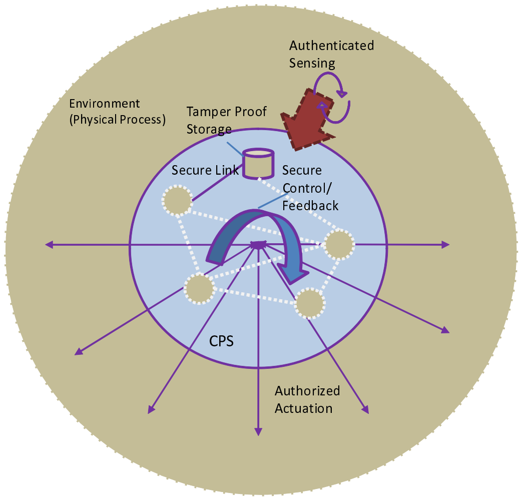


Fig. 3. CPS Security Requirements

along with the environment knowledge/ information to function. The idea is to use the monitoring capability of CPSs to provide security. By utilizing this fundamental capability of CPS, security provision becomes intrinsically linked to CPS operation and not something that is added to a functioning system to protect it from threats. Another merit of CYPSec solutions is that they can now harness the complex and dynamic nature of the physical process for security purposes. Figure 4 shows the principal characteristics of the CYPSec security solutions. Some of the principal characteristics of CYPSec solutions are:

- *Usability*: As CPSs are closely linked with their environment, changes to the environment have a direct affect on the system and this can be utilized in the security realm as well. By using environment characteristics as a basis for security primitives, security deployment and management abstractions need not be considered freeing the designers to focus on functional aspects of the system.
- *Emergence*: CYPSec solutions are designed to not only provide the appropriate security functions for which they are designed for example confidentiality, integrity, availability but also demonstrate additional "allied" properties, such as authentication and adaptivity.

- *Multi-domain Primitives*: As CYPSec solutions have both cyber and physical aspect to them, enabling them usually requires integration of techniques from multiple domains with security. Further, as the solutions are used to work in tandem with existing infrastructure, CYPSec solutions should be implementable with well defined computational primitives.

It needs to be pointed out that, even though CYPSec solutions address many of the challenges of CPS security (meeting application specific security requirements and usability of security), its dependence upon the environment may make CYPSec solutions susceptible to the physical threats on CPS. However, this may not make it completely unusable, as for many domains the environment may be naturally secured. A case in point is a health monitoring CPS deployed on a patient's body. CYPSec for such systems have used physiological signals from the human body as a means of addressing their security needs. However, the human body is one of the more secure environments in which CPSs can be embedded. It is not easy to alter its functioning, at least in a surreptitious way. However, the same cannot be said for power-grids where many substations are unmanned. Therefore, if the physical process itself is not secure, some mechanism for authenticating the sensed value is required. However, most of the examples of CPS fall somewhere in between the two in terms of physical process security, as in the case of CPSs deployed in smart data-centers [121] for reducing its cooling cost. Attackers can potentially tamper with the sensors in a datacenter requiring the air conditioner to overload, but this would require physical access to the datacenter itself, a non-trivial task.

1.3. Overview of Research Problem and Contributions

In this section we discuss the principal contributions of the dissertation. Principally, we present CYPSec solutions for two different, but related problems - securing communication within a Body Area Networks (BAN) on Pervasive Health Monitoring Systems (PHMS), while the second involves developing adaptive and proactive access control model in a smart-infrastructure especially during emergencies. We chose the two application domains as they are: 1) good representations of cyber-physical systems given their environment-coupled nature, use of low capability sensors for monitoring their environments and networked operation; 2) they demonstrate the broad applicability of CYPSec solutions from sensor level to system level security; and

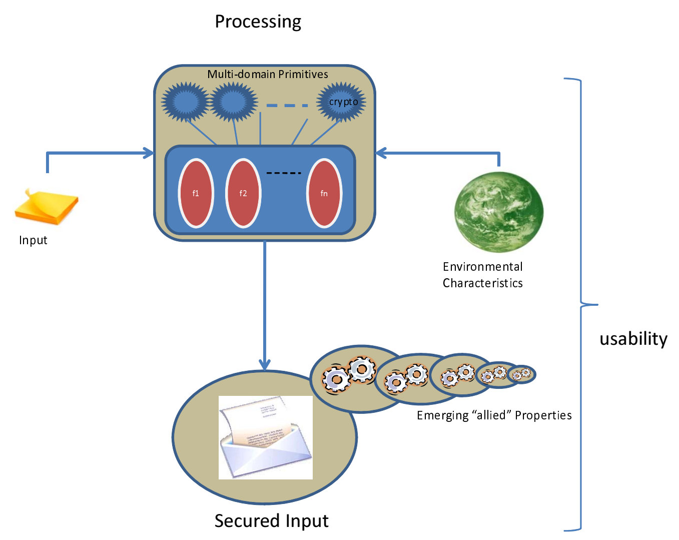


Fig. 4. Cyber-Physical Security Solutions: Conceptual Overview

3) they demonstrate the two variations of the altered threat model for CPSs, with BANs exhibiting complete physical security, while smart-infrastructures exhibiting physical security to much lesser degree.

1.3.1. Securing Body Area Networks in Pervasive Health Monitoring Systems (PHMS)

Importance of Problem: Pervasive Health Monitoring Systems are used to provide continuous real-time monitoring of a person's health. PHMS' gather health data using a network of medical and ambient sensors/devices worn by the host (patients). This *Body Area Network* (BAN) of sensors collects data from patients and uses wireless interface to forward them (potentially through other sensors) to higher-tier entities in the system. The wireless channel, if left open, will allow any entity in the vicinity of the patient, to eavesdrop on the data being communicated. In order to secure this communication, each time the network is setup, each sensors has to be programmed with a secret key with their neighbors (symmetric key crypto-systems used here due their inherent simplicity). This process could be executed by the doctor during the deployment of the sensors on the patient, or by the patient themselves. As PHMS become more commonly accepted and worn by people with no chronic ailments (e.g. fitness monitoring), the overhead imposed by pre-deployment - in terms of security, user involvement, and general network management - will affect the basic patient monitoring functionality of the system, and increasingly at odds with the push toward *usable* (plug-n-play and transparent) and *interoperable* medical sensors [135] [136]. All sensors provide the constructs to communicate securely, e.g. encryption, decryption and hashing modules, the real overhead for secure inter-sensor communication is key distribution. *Therefore, to enable secure communication and data collection in BAN, we need to develop a key agreement techniques which preserves the usable character of the health monitoring infrastructure.*

Overview of Solution: We have developed a novel CYPSec solution for securing the communication in a BAN called the **Physiological Signal based key Agreement** (PSKA) [23] [127]. PSKA is designed to enable automated key agreement between sensors in the BAN without any form of external user involvement. The establishment of the key establishes the trusted channel between the sensors in the network. It functions by generating simple frequency domain (Fast Fourier Transform (FFT) based) feature vectors from common physiological signals (e.g. Photoplethsymogram and Electrocardiogram) from the human body, and using them to hide/lock a session key in a cryptographically secure manner. The hidden session key is then transmitted to another sensor in the BAN in the open, which uses its own copy of the feature vector (derived from the same physiological signal in a loosely synchronous manner) to un-hide/unlock the session key. PSKA provides communication security in a health monitoring CPS. Its functioning inherently depends upon features from its environment. It takes advantage of the deeply embedded nature of sensors in BANs and their ability to monitor specific physiological signals from their hosts. The result is a security solution that *combines signal processing along with cryptographic primitives* for secure key agreement. Using PSKA, nodes in a BAN do not require any additional keying material to secure inter-sensor communication, and sensors can now be strapped on by a patient to ensure secure communication in a usable (plug-n-play, largely transparent) manner. A malicious entity cannot eavesdrop on the communication unless it has access to physiological signals. It can be seen that, apart from establishing a secure channel between sensors in a BAN, PSKA's brings out many new *"allied" properties* to security which traditional (mostly involving manual key pre-deployment) secure key agreement protocols do not posses: 1) *Secure Interoperability :* The PSKA CYPSec solution reduces one of the important hurdle from achieving CPS interoperability by enabling entities in a CPS to establish a secure channel between one another enabling them to communicate in a seamlessly manner; 2) *Authentication :* As physiological signal features uniquely represent the host at that given time, any successful agreement of keys between two sensors in a BAN assures the participating nodes that the other is in the same network. This eliminates the need for additional authentication primitives needed by traditional key agreement protocols such as Diffie-Hellman and its variants. We have implemented the protocol on Matlab and Crossbow mote platform and have analyzed its performance in terms of security and energy-efficiency and sustainability.

The principal contributions of this work are:

- **Characterization of BAN Security Issues:** The principal properties of securing inter-sensor communication in BANs in a PHMS were identified along with problems of using traditional approaches to address them [130].

- **Cyber-Physical Security Solution:** A secure, usable (plug-n-play, transparent) key agreement protocol (PSKA) was proposed which uses physiological signals based features for hiding and un-hiding the keys during distribution [127].

- **Implementation:** A benchmark implementation of PSKA was done using Matlab to illustrate the correctness of the scheme. Further, a Crossbow mote based prototype was also successfully implemented on the *Ayushman* - a PHMS test-bed [125]. Photoplethysmogram and Electrocardiogram signals, collected from real subjects, were used for the implementation.

- **Performance Analysis:** Both the PSKA implementations were analyzed in terms of level of security they provided and the properties of physiological features used. The comparison of the benchmark and mote implementations demonstrated the viability of the protocol on resource-constrained platforms such as motes in terms of computation, communication and memory requirements [125]. Further, a study of the energy characteristics of the mote implementation showed PSKA's *sustainability* for various energy scavenging techniques [128].

1.3.2. Securing Information Access in Smart-Infrastructures

Importance of Problem: Smart-infrastructures are becoming increasingly common with the embedding of computational capabilities into mundane entities surrounding us. Such an infrastructure has the ability to monitor itself and its users and provide context-aware information and services to them. Applications such as the active spaces [112] and pervasive health monitoring systems [113] [133] [129] are examples of this trend. Smart-infrastructures due to their "self-aware" nature can be used to detect, provide useful and real-time information about the state of such emergencies to the planners and relief-workers and facilitate response, thereby improving the chances of saving lives and property. Given the amount of sensitive information available in such systems, in order to prevent a malicious entity from accessing information or services from such a system in an unauthorized manner, access control schemes are used. However, during emergencies, traditional access control models are disabled to enable emergency management. Such an approach is ill-advised as it allows anyone to simulate an emergency and get open access to it. *Therefore an access control model is needed specifically designed for smart-infrastructures, that not only has the ability to control access to patient data in normal situations, but also dynamically adapt its behavior to handle critical/emergency situations in an effective manner*.

Overview of Solution: For securing access to information in a smart-infrastructure, we have developed a novel access control model called **Criticality Aware Access Control** (CAAC) [44]. CAAC is a CYPSec access control model which has the ability to provides the right set of privileges for the right set of subjects, at the right time for the right duration to facilitate emergency (*criticality*) response. *Criticalities* are situations which require urgent *response actions* in order to *control* and maintain the stability of the system. In practical terms, controlling a criticality means eliminating the loss of lives and property in the event of a criticality, and the response actions are designed to achieve this goal. Each criticality has a timing duration associated with it known as *window-of-opportunity*, within which response actions have to be taken for the criticality to be controlled [44].

CAAC is an adaptive access control approach designed to facilitate the control of all the active criticalities within the system. It uses an *Action General Model (AGM)* based on the stochastic crisis planning technique developed in [83]. The results of AGM execution (list of response actions for different combinations of criticalities such that the window-of-opportunity of all the criticalities is satisfied) are provided to the CAAC, before it is deployed. CAAC monitors the environment it is deployed in, at regular intervals, and depending upon the state the system is in (the current state of the system is a result of all responses and criticality which occurred since the system observed its first criticality), it identifies the best response actions (authorizations) that need to be taken from that state to reach the normal state. It then identifies the set of subjects (which can be either statically specified or dynamically computed based on subject context) best suited to execute the response actions, and provides them with credentials to execute the actions. We call these new privileges for chosen subjects as *alternate privileges*. This change in credentials of subjects is temporary and reverts back when the criticalities in the system change, are controlled or expire.

CAAC enables authorized actuation in a CPS, and can be used in both passively active and active modes, as it only facilitates the execution of response actions. CAAC, *combines stochastic crisis model with access control primitives*, and enable authorizations of subjects of the system for managing criticalities even before they explicitly ask for it, making emergency management more user-centric. CAAC inherently also brings out a new *"allied" properties* to security policy specification: 1) *Proactivity* - The ability to provide privileges to subjects in response to emergencies without explicit request; 2) *Adaptive Risk Management* - Being aware of its environment, CAAC minimizes the risks associated with voluntarily opening the system, by making the opening temporary, for a specific set of subjects only, and dynamically varying it over time as criticalities within the system change. The principal contributions of this work are:

- **Characterization of Data Access during Criticalities:** The principal need and issues involved in data access during emergencies/criticalities were studied [44] [83]. A set of design goals for criticality aware system were identified.

- **Adaptive and Proactive Access Control:** Developed a novel adaptive and proactive access control model (CAAC) which can provide appropriate authorizations to subjects during emergencies without explicit requests [124].

- **Case Study:** A detailed case study was done on implementing CAAC on a smart-infrastructure (smart oil rig) to demonstrates its capabilities.

- **Implementation:** A prototype of CAAC was implemented as part of the the Ayushman PHMS Test-bed for validation [124], and shown to satisfy the design goals.

1.4. Intellectual Merits

In this dissertation, we propose that the cyber-physical security solutions provide a novel way of looking of securing systems which are deeply embedded in their environment. By using features from the environment as a integral part of a security solution, they can be made easy to use, proactive and adaptive to the changing requirements of the system. Some of the principal contributions of this dissertation are:

- Proposed a novel security solution for BANs which combines physiological signal processing and cryptographic primitives for securing inter-sensor communication.

- Solved the challenge of implementing the computationally complex signal processing and mathematical routines such as Fast Fourier Transforms, and signal peak detection and polynomial convolution on resource-constrained Crossbow mote platform. Demonstrated the security and sustainability of the solution based on well known energy-scavenging techniques.

- Proposed a proactive access control model for smart-infrastructures which can facilitate automated emergency management by temporarily providing the required privileges to users without their explicit request.

- Developed a stochastic emergency management model (which took into account the probability of criticalities and correct execution of potential response actions) for improved adaptiveness - determining and enabling the best set of response action for the set of criticalities within the system.

- Developed a pervasive health monitoring, called *Ayushman*, test-bed by integrating diverse sensing, medical monitoring, processing and communication platforms. Ayushman was used for implementing the proposed solutions for validation and performance analysis.

The methodology and approach used in this dissertation would bring about radical improvement in the design of the secure, usable, energy-efficient and sustainable Cyber-Physical Systems. The anticipated application areas include interoperable medical devices, smart-homes, care-facilities, vehicular networks, and any smart user aware system.

1.5. Dissertation Outline

The rest of the dissertation is organized as follows: Chapter 2 we present our solution for securing BANs in a PHMS - Physiological Signal based Key Agreement (PSKA). We present details of the principal issues in utilizing physiological signals for security purposes, present details of the protocol, validate it through a proof-of-concept implementation and analyze its performance. In Chapter 3 we present Criticality Aware Access Control (CAAC) our adaptive and proactive access control model for securing data access in smart-infrastructures in the event of emergencies or criticalities. We present details on the need for and the workings of the access control model, and a detailed case study in using it for larger applications. Chapter 4 presents the Ayushman test-bed which is used to implement prototypes of PSKA and CAAC. Lastly, in Chapter 5 we present some concluding remarks on our work and prospects for future work.

2. BACKGROUND

In this chapter, we present some general concepts and definitions pertaining to the notion of *security* which we use throughout the dissertation. Though the notion of security has many connotations, for the purposes of this work, we define it as *preventing unauthorized entities from viewing or modifying data generated within a system*. We use the term system (in this chapter alone) in a generic sense to mean a computing system which takes an input, processes it and provides an output. We begin this chapter by defining the requirements of security, the potential attack vectors, the approach to addressing them and how CYPSec can be used in this enabling these approaches.

2.1. Security Requirements

In order to secure any system we have identified certain requirements which need to be met.

- *Data Integrity*: Ensures that all information generated and exchanged during the system's operation is accurate and complete without any alterations.
- *Data Confidentiality*: Ensures that all sensitive information generated within the system is disclosed only to those who are supposed to see it.
- *Authentication*: Ensures that the system knows the identities of all the entities interacting with it, and vice-versa.
- *Authorization*: Ensures that any entity trying to access particular information from the system is able to access only that information which they are entitled to.
- *Availability*: Ensures that any entity which uses the data and services and resources of the system are able to do when required. In the context of this work we are not addressing it as the primary requirement as we assume that it is always satisfied.

2.2. Potential Threats

A *threat* is a violation of security [15]. A system needs to be guarded against them, in order to ensure its correct operation at all times. The execution of the threats is called an *attack* while the entities which execute these threats are called *attackers* or em adversaries. We classify the common threats into four categories:

- *Eavesdropping*: The attacker can intercept any information communicated by the system. It is *passive attack*, i.e. the attacker does not interfere with the working of the system and simply observes its operation. Systems with communication semantics are particularly susceptible to eavesdropping through traffic analysis. Confidentiality security property ensures that it can be avoided.

- *Spoofing*: The attacker poses as a legitimate part of a system and tries to take part in its operation. Once mounted successfully, the attacker can not only access information from the system, but also modify or delete it, apart from introducing incorrect information. It can also result in incorrect execution of and delay in provisioning of specific services. At any point during the attack the system is unaware of the deception assumes all the data and services received from the attacker as legitimate. Attack vectors such as man-in-the-middle are a variation of this class of this class. As the attacker interferes with the working of the system, it is an *active attack*. Authentication, authorization along with confidentiality and integrity protection are needed to stave of this attack.

- *Denial of Service*: The attacker inhibits the basic operation of the system. The attacks are active in nature and can be mounted for example by interfering with its functionality (e.g. jamming), overwhelming it (e.g. by providing a bogus large input set to be processed causing battery depletion or rejection of legitimate inputs) or through physical compromise (e.g. modify the hardware/software of specific parts of a system resulting in its inaccurate operation). The availability security property ensures that it can be avoided. We have included this attack for completeness sake but do not actively consider it in this dissertation.

2.3. Security Approaches

In order to eliminate the effects of threats we have to ensure that the security requirements are satisfied. In this section we present a brief overview of the techniques involved in ensuring the four principal security requirements are satisfied.

2.3.1. Confidentiality

Confidentiality requires the ability to hide data. This is usually accomplished using a cryptosystem. A cryptosystem is a mathematical function which transforms (encrypt) an input value, called plaintext, into a garbled form, called ciphertext. The ciphertext can only be transformed (decrypt) back to the plaintext using an inverse function. Both encrypt and decrypt operations require the knowledge of a secret called the *cryptographic key*. Without knowing the exact value of the key used in encrypting a plaintext, it is computationally hard to decrypt the message. There are two types of cryptosystems that can be used for confidentiality maintenance - *symmetric* and *asymmetric*.

Symmetric cryptosystems are those in which the entities performing the hiding and un-hiding share a common key. It assumes that the entities which has a knowledge of a key has the right to view the information encrypted using the key. Therefore, if the key is stolen, lost of guessed by an attacker, then the confidentiality property is lost. Some of the prominent algorithms which use symmetric key are AES [29], RC5 [103], Blowfish [109]. Asymmetric cryptosystems use a pair of cryptographic keys, public key and private key. The keys have the property that, data encrypted with one key can only be decrypted with the other. Each entity which wants to communicate securely is assigned a pair. The public key of an entity is published and well known and is used for encryption by any others who want to communicate securely with it. The private key is secret and is used to decrypt any messages received from the others. Asymmetric cryptosystems are mathematically intensive and are comparatively expensive to their symmetric counterparts. Therefore, they are primarily used for exchanging symmetric keys. Prominent algorithms which use key pairs are RSA [101], Diffie-Hellman [31], ElGamal [37].

2.3.2. Integrity

Ensuring data integrity requires the ability to detect any changes introduced (maliciously or otherwise) in the message being communicated. This is usually done using a message digest - a one way (hash) function which takes as input the data whose integrity is being protected and produces a fixed length random value called a digest. As the function is one-way in nature, given the digest it is computationally infeasible to re-create the input, further even a on-bit change in the input produces a drastically different output.

Typically the use of message digests is linked to the cryptosystem being used, for the symmetric case, a digest is generated using a *Message Authentication Code* (MAC) - a function which takes as input data being protected, and the shared symmetric key. At the other end, a MAC is computed on the received data using the shared key, if the received MAC and the locally computed MAC match, the data is assured to be non-tampered. The key is used in order for computing the digest because without it a malicious entity could potentially replace the data and its digest with its own value and appropriate digest. For the asymmetric scenario, a simple one-way hash function is used to obtain the digest of the data which is then encrypted using the sender's private key; this is called a *digital signature*. The entity which wants to verify the data integrity, computes a hash of the data received, uses the sender's public key to decrypt the signature compare the decrypted hash with the locally computed one. If they are same, then data integrity is assured. Prominent cryptographic digest algorithms are MD5 [102] and SHA [51] and their more secure variants. The same algorithms can be used for MAC purposes as well by changing their execution slightly to include the symmetric key as one of the inputs.

2.3.3. Authentication

Authentication is the binding of an identity to a principal [15]. It establishes a level of trust between the two entities which then forms the foundation of all subsequent communication. In interactive systems, authentication assures the communicating entities the identity of the other. Both entities usually have to provide some form of identification credentials which the other verifies to confirm the presented identity. The form of identification presented varies from situation. Some of the well known authentication techniques include digital certificates, biometrics (fingerprint, iris scan, voice), challenge-response interactions (passwords, smart-cards, RFIDs) [108] [15]. Many-times a combination of authentication techniques are used; this is known as multi-factor authentication. Multi-factor authentication is required because of the probabilistic nature of authentication. No credential can prove with certainty the identity of the presenter as it is always possible some malicious entity is presenting the identity in lieu of the legitimate entity. By repeating the authentication process using different mechanisms multiple times increases the probability of that the entity is who it claims to be.

2.3.4. Authorization

Given the identity of an entity interacting with the system, authorization identifies and manages the system's data and service objects available to them. The primitive that is used for managing authorizations is called an access control model. The access control model is defined for each object within the system. In its basic form it works as follows: 1) an entity which wants to use an object within the system makes a request, 2) the access control model takes the identity of the entity and assigns identifies the privileges available for it based on well defined rules, and 3) if the request matches the privileges then access is allowed else it is denied. Privileges are tokens which define the rights that the entity has on using the objects available within the system. Access control models can be classified into three categories, depending upon who defines them and how are they organized. Discretionary Access Control (DA) rules are defined by the owner of the data or service object within the system. Mandatory Access Control (MA) rules are defined by the system administrator and not the owner of the individual objects. Both these models use identity of the entity access the system as the basis of providing privileges. DA and MA are typically implemented using Access Control Lists (ACLs). ACLs are id and privileges tables maintained for each object in the system, which given the identity of the entity trying to access the system, determines their privileges on that object [15].

Another access control technique called Role based Access Control (RBAC) takes a different approach. RBAC is an administrator-defined access control model which bases its decisions on privileges based on the function of entity performs in the system rather than their identities. It assigns roles to each entity interacting with system and then defines privileges for each role. This makes it easier to manage the entities access the system and the privileges available to them, compared to DA and MA where one has to maintain a list of all the subjects and privileges. It has been shown that RBAC is general enough to implement the features of both DA and MA [106] and will be used as the principal access control model in this work.

2.4. CYPSec Solutions

In this dissertation work, we demonstrate the use of CYPSec solutions of ensuring four of the fundamental security requirements/properties - confidentiality, integrity, authentication and authorization. From the discussion above we can see that for confidentiality and integrity protection, the fundamental requirement is a cryptographic key. Key distribution is therefore one of the pillars of security itself. We have developed, Physiological Signal based key Agreement (PSKA) - a CYPSec secure key agreement solution, to demonstrate how environmental characteristics (physiological signals) can be used for the agreement of symmetric keys between two sensors in a BAN. Further, an additional requirement for key distribution is authentication of the distributer. This is usually carried out by additional authentication mechanisms such as manual pre-deployments of the keys or use of digital certificates in the case of using asymmetric cryptosystems for key distribution. With PSKA, we do no need additional steps for authentication, the successful key agreement automatically ensures authentication as well. For demonstrating the ability of CYPSec solutions to enable authorization, we look at Criticality Aware Access Control (CAAC) - an environmentally coupled access control model, which implements a proactive access control model for emergency management in smart-infrastructures. It uses constructs of the RBAC model, a stochastic model and the knowledge of the current state of the system to enable appropriate privileges for subjects in the space for managing emergencies, without an explicit access request.

3. SECURING BODY AREA NETWORKS IN PHMS

3.1. Introduction

The health-care system in developed countries is coming under increasing pressure as a result of the dramatic increase in the population of the elderly and without a corresponding increase in trained medical staff [7]. In response, recent years have seen considerable work on the development of **Pervasive Health Monitoring Systems** (PHMS). A PHMS consists of large number of tiny wearable and implantable wireless sensors and medical devices (collectively called *sensors*) which collect physiological and environmental data from the patients. The data collected by the sensors is forwarded through the wireless medium to a sink entity called the *base station*. The entire collection of health monitoring devices and the sink is called a *Body Area Networks* (BAN). The base station itself forwards the data to more capable entities which provide facilities for organizing pre-processed health data into a structured electronic format (also known as *Electronic Patient Records (EPR)*), and storing them for the long-term, and providing access to the caregivers when needed [53] [113] [133].

PHMS provide a proactive patient-centric form of care (as opposed to the reactive caregiver-centric form of today's care) which permit caregivers to monitor their patients at all times, enabling detection and response to health problems as soon as they occur. In a world where the population is growing at a rapid rate, medical institutions are facing severe shortage of trained medical staff, health-care costs are skyrocketing and incidences of medical errors are at a all time high [117] [5], PHMS can play a major role in alleviating these problems. Some of the examples of applications where PHMS can be utilized include: 1) mobile telemedicine, 2) disaster response, 3) hospital-based care, 4) lifestyle management [123].

As PHMS is used to collect sensitive information (health data), it has to be able to protect it from entities which have no right to see the information. Security solutions are therefore needed within its two planes to prevent any form of unauthorized access and preserve *patient privacy*.

3.2. Preliminaries

The term *Privacy* is defined as the ability to allow only the intended recipients from being able to view a given information [56]. Privacy preservation is of particular importance in systems which collect health related information. A person's health information is very sensitive in nature. Exposure of such information in an uncontrolled manner can lead to misuse, such as: 1) deliberate alteration of health data of specific patients, leading to wrong diagnosis and treatment, 2) deliberate generation of false alarms or suppression of real alarms raised by the system in case of emergencies, and 3) economic and social discrimination of patients (insurance companies offering health insurance high premiums to people who have certain chronic problems).

3.2.1. Motivation for PHMS Security

With the electronic nature of data and wireless communication within PHMS, the threats to patient privacy are much more severe compared to today's largely paper based records. The reasons for these vulnerabilities are [132]:

- With paper based system, health data is collection is a manual process with the caregiver copying the patient's medical stats on to the patient record. In the case of PHMS patient data is collected automatically and transmitted through the wireless channel in the open which makes it easier to access for everyone including malicious entities.
- Paper records are stored in highly centralized locations and any copying of this information is tedious and time-consuming process. PHMS on the other hand maintain electronic records on networked systems (for availability reasons) which are again easy to access and copy from anywhere in the world.
- More and more sensitive information, such as HIV status, psychiatric records and genetic information, is being included in a patient's medical record for faster and easier retrieval.

Therefore the ability of PHMS to continuously collect, exchange, store electronic medical data presents many avenues of abuse of privacy. Indeed preserving patient privacy is also a legal requirement as per *Health Insurance Portability and Accountability Act (HIPAA)* which mandates all personally identifiable health information be protected [48]. Developing security solutions for privacy preservation in PHMS is therefore paramount.

The need for security solutions for PHMS has been well documented [113] [45]. However, most PHMS considering their time-constrained *mission critical* nature [126], are still designed solely to monitor health conditions with the goal of achieving data availability, efficiency, and self-configuration [45]. Any user (and the environment itself) is considered benign and security is mostly dispensed with. Such an approach can have serious consequences especially as the technology becomes more wide-spread. A case in point is the recent news on pacemakers which were shown to communicate sensitive data in clear channel and were easily reprogrammed by researchers posing as potential attackers [28]. This peripheral treatment of security in health monitoring systems occurs because adding security tends to make the system: 1) difficult to implement, 2) complex which makes it difficult to manage and debug, 3) difficult to use for the patients and caregivers, and 4) expensive not only in terms of costs but also computationally and energy-wise [45]. However, we believe that considering security, from the start, can address many of these issues.

3.2.2. PHMS Security Requirements

In order to secure PHMS, *securing the data transmission process in its BAN is paramount*. The BAN is the least capable part of the entire system, and therefore a potential weak-link. Lack of security in the BANs may not only lead to loss of host's privacy, but may also result in physical harm. For example, adversaries can introduce bogus data in the network or modifying/suppressing legitimate ones, inducing erroneous diagnosis. Designers of BANs therefore have to ensure that adversaries do not: 1) join the network pretending to be a legitimate node, 2) access confidential health data collected or exchanged within the BAN, and 3) preventing some or all health data from being reported or modifying it. One of the most vulnerable aspects of BANs is the wireless medium used by the sensors to communicate their data. This allows adversaries to not only eavesdrop on the messages exchanged (using a receiver tuned to the carrier frequency), but also capture and modify the messages being communicated.

Given the requirement of privacy in PHMS and the many threats that such systems face, the need for developing security schemes for PHMS is apparent. Fortunately, the requirements of security schemes for PHMS is not very different from traditional systems and relies on the maintenance of three basic properties and one additional one:

- **Data Integrity**: All information within PHMS is collected and transferred in an accurate and complete manner without any alterations in any way.

- **Data Confidentiality**: Information in PHMS is only disclosed to those who are authorized to see it.
- **Authentication**: Ensure correctness of claimed identity of communicating entities.
- **Usability**: Though not explicitly considered for security solutions, we believe for PHMS environments usable security solutions - those that are *plug-n-play* in nature (do not require user involvement in initialization or maintenance) and *transparent* to its users.

3.2.3. Traditional Solutions and their Pitfalls

Secure communication between sensors is well-understood and essentially has two phases: *trust establishment* and *data communication*. Trust establishment is a means of assuring the communicating entities that the other is legitimate. It is usually carried out through the establishment (and use) of cryptographic keys between the communicating entities. Once the key (for the purposes of this discussion we assume symmetric keys, without loss of generalization) is established between two sensors, proving the presence of the key to the other establishes trust. With the trust established, data communication can now commence, which also utilizes the same key for maintaining the confidentiality and integrity of any data exchanged.

Key establishment (also referred to in the literature as key distribution or agreement) between sensors is therefore one of the most important aspects of secure communication between sensors. Traditional (non-BAN) symmetric key distribution schemes for sensor networks can be classified into three generic categories:

- *Solely Pre-deployment based*: These techniques require storing large set of keys in each sensor node before deployment. Pre-deployment can be accomplished in a deterministic manner if the physical deployment details are known. For example, each node shares a pair-wise key with all its neighbors, a group key for multi-casting to its neighborhood and a network-wide key, to meet its communication requirements. However, given the long term monitoring requirements of the BAN and static nature of the keys, exposure of the pre-deployed keys to cryptanalytic attacks may also pose a problem. Another pre-deployment mechanism is to store a large set of keys, derived from a common key pool at each node, such that finding a common key between any two sensors is high. These schemes have a large space requirements. Prominent examples include - [39] [20] [35] [67] [91].

- *Communication-based*: These techniques require some form of communication between entities in the network for key distribution. Such schemes either obtain keys from a central entity like the base station or require the exchange of information such as random numbers or node IDs, which along with a pre-deployed *master key*, is used for generating the shared key. Examples include - [142] [63] [89].

- *Public key cryptography based*: These techniques use a pair of related keys (public and private) along with mathematically complex and computationally intensive algorithms to distribute symmetric keys between sensors. Prominent examples include - [74] [50].

It can be seen that each of these protocol classes assumes some form of *pre-deployment*, which itself requires an underlying assumption of trust. This pre-deployment acts as an initial source of trust and forms the basis of distributing the actual key to be used in secure communication between sensors. However, this assumption makes them unsuitable for BSNs. The main reasons for this stem from the following insecurities and hinderance to usability associated with pre-deployment:

- If the pre-deployment takes place at the factory where the sensors are manufactured, then hosts cannot trust the keys unless the entire key distribution chain from the factory to the host is secured [60].

- If the pre-deployment is to be executed by the host, it would require them to make important decisions about the keys to be used. This might result in poor quality keys; adversely affecting the security of the system.

- With pre-deployment adding or moving nodes within the network would require additional host involvement. For example, if a node is added to the network, all the nodes in the neighborhood have to be have their keys updated.

- With pre-deployment, it is very difficult to change the keys in the network which have been compromised.

Over the years some solutions have been proposed for secure pre-deployment such as Message in a Bottle [60] or Resurrecting Duckling [116]. These techniques expect the presence of additional equipment

(Faraday cage) or extra communication interfaces on the sensors for a side-channel key deployment (for example infra-red, ultra-sound), and more importantly these techniques involve considerable user involvement - making them unsuitable for BANs. Public key cryptography on the other hand works slightly differently. It does not require any key pre-deployment which is ideal for BANs, however, the principal problem comes in trust establishment itself, as there are no secret between them. Additional authentication mechanisms are needed which in order for this to work, which may require host involvement making their behavior akin to pre-deployment based scenarios discussed above.

As BANs become more commonly accepted and worn by people with no chronic ailments, for preventive or non-medical applications (e.g. fitness monitoring), the overhead imposed by pre-deployment - in terms of security, user involvement, and general network management - will adversely affect the **usability** of the system. *We need a forward-looking security solution which follows the "plug-n-play" paradigm, is transparent and easy to use which not only meets today's needs but is also flexible enough to be used in the future.*

3.3. Our Approach

To overcome the issue of making security protocols more usable, we propose a CPYSec solution - *Physiological Signal based Key Agreement* (PSKA). PSKA takes a fundamentally different view of security by *combining signal processing with cryptographic primitives* as a part of their solution. In our case, it enables automated key agreement between sensors in the BAN without any form of user involvement by utilizing specific physiological stimuli from the human body. PSKA's *environmentally-coupled* key agreement means nodes do not require any additional keying material to secure inter-sensor communication. Sensors can now be strapped on by a patient to ensure secure communication, in a *plug-n-play* manner. By extension, it also allows adding, moving, or removing nodes without having to re-key sections of the network. *This is achieved by utilizing the dynamic nature of the human body which produces chaotic, distinctive and time variant stimuli. The stimuli characterizes the very specific situation the subject is in, and therefore makes it difficult to predict current values even if past values are known [138].* We presented a detail discussion on the goals of PSKA,

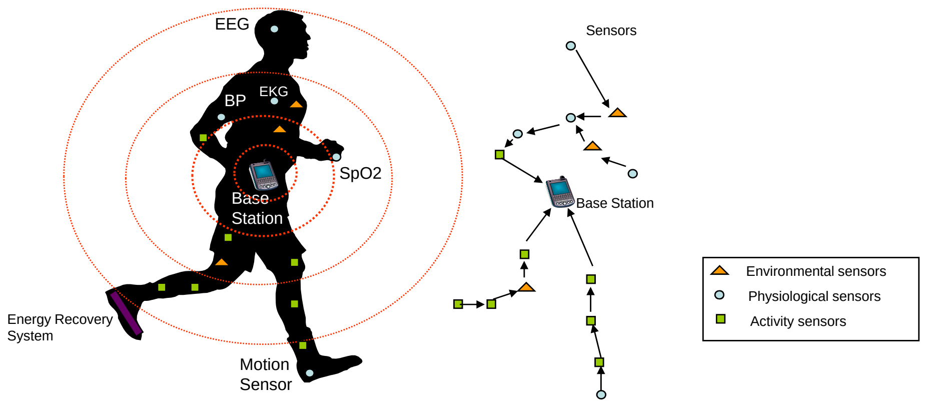


Fig. 5. A Health Monitoring Body Sensor Network

the issues involved in implementing it, its security capabilities and performance results. But, before we delve into these details, we begin with the system model which presents our assumptions regarding the BAN.

3.4. System Model

A BAN is health monitoring network of sensors, implanted or worn by a person called the *host*. The sensors in the BAN are *heterogeneous*, possessing the ability to measure multiple stimuli[1] from the host's body. They are assumed to be built to survive extreme conditions like variation in temperature and presence of water [86]. We assume the sensors *communicate through the wireless medium*, as wires running between sensors in a BAN will make it obtrusive specially in the case of implanted sensors or when there is a need to reconfigure the placement of sensors on the body.

It has been suggested in [61] that a large number of low quality sensors can perform the task of monitoring as effectively as a few high quality sensors. Such a network will have multiple sensors measuring the same stimuli providing redundancy and therefore better fault tolerance, while at the same time being *less intrusive*, *energy-efficient*, and conducive for regular wearing due to their *light weight and small form factor*. Networks with sizes of up to 192 and 255 have been already been proposed in [57], [24] respectively. The BAN is therefore assumed to be a potentially dense network consisting of nodes numbering in the hundreds.

[1]Already physiological monitoring sensors are becoming multi-modal and are being able to sense multiple types of stimuli [61] [84].

A *base station* is used to collect data from the entire BAN, as it has significantly higher computational and communication capabilities compared to the sensors (see Figure 5).

The sensors once deployed are static (no movement) in nature, but can be adjusted as and when required[2]. Moreover, as the sensors are expected to work for long time durations (in some cases the entire lifetime of a patient), such adjustments may be needed from time to time to ensure correct operation of the BSN. This ability to control/manage every aspect of sensor deployment makes them fundamentally different from traditional sensor networks. Nodes in a BSN can be managed at multiple levels unlike traditional sensor networks:

1. *Node level*: Nodes have to be placed such that they measure various physiological values accurately enough to communicate. For this reason, the physical packaging and placement of nodes and any contact they have with the host's body can be manipulated to minimize noise and measurement artifacts.

2. *Link level*: For each node, its ID, location of deployment, physiological signals it can measure, and IDs of all the nodes within its range, are known and can be controlled.

3. *Network level*: Sensors in the BSN may have to undergo *adjustments* from time to time (changing sensor contacts and interface with the host and so on) along with *re-organization* (changing faulty sensors, adjusting sensors location, contact and placement, addition of new improved sensors and removal of faulty, old sensors) to ensure correct operations over long-term deployments.

Link level control is provided to some extent by traditional sensor networks, where details about the nodes and their immediate neighborhood is assumed to be known. The latter two, however, are unique to BSN.

3.4.1. Threat Model

BANs potentially face many *threats*, due to the sensitive nature of the data they collect, and the broadcast nature of the wireless medium they use to communicate. The threats originate from two sources: *active* and *passive adversaries*. *Active adversaries* have the capability to eavesdrop on all traffic within a BAN, inject

[2]We believe, the assumption can be easily relaxed without considerably changing our scheme

messages, replay old messages, spoof, introduce or compromise nodes in the BAN in order to become part of the network. They may also try to use the physiological signal data obtained from other people to compromise the BAN. Active adversaries, if successful, can not only invade a patient's privacy but can also suppress legitimate data or insert a bogus one into the network leading to unwanted actions (drug delivery) or prevent legitimate actions (notifying doctor in case of an emergency). *Passive adversaries*, on the other hand, are adversaries which eavesdrop on the message exchanged within the BAN and use off-line cryptanalytic attacks to access any confidential data being communicated thereby invading a patient's privacy. They do not try to interfere with the functions of the BAN. Therefore, maintaining confidentiality, integrity and authenticity of the communicating entities against active and passive adversaries is of paramount importance in a BAN.

It needs to be pointed out that we do not consider Denial of Service (*RF jamming, electro-magnetic interference, or battery depletion*) attacks on BANs in this work. We assume that the intensions of the attacker is to subvert the system in a clandestine manner, and not to simply prevent it from functioning as this can be noticed immediately by the host or the administrator and appropriate action can be taken to overcome it. Further, we assume that adversaries are not in contact with the host's body. *We do not consider the compromise of the physical process or the BAN node itself in this work.* A physically compromised node would have to be in intimate contact with the host. Although this is possible it is unlikely due to various logistical reasons - it is not easy to insert a node inside a human body without the knowledge of the host or the host's surgical team (which is trusted) during an operation, and anything worn is mostly under supervision of the host (if the host is not capable then under the supervision of the caretaker).

3.4.2. Trust Model

To address these threats, we assume the following *trust model* for BANs: The wireless medium is not trusted. It was recently demonstrated by researchers from University of Washington and University of Massachusetts, Amherst that an implanted defibrillator could be hacked to reveal the patients health data as well as administer an untimely shock [45]. This attack was possible due to the open wireless channel used by the device to communicate with its programmer. Such an attack can be easily extended to BANs where sensors collect patient data, actuate treatment and perform wireless communication between one another to relay patient data to the base station. The sensors do not accept any message they receive unless they can authenticate the sender. As physical tampering is not possible, the sensors trust each other and the base station.

3.5. Physiological Signal based-Key Agreement (PSKA)

In order to meet the requirements of securing BANs, we present **Physiological Signal based-Key Agreement** (PSKA). PSKA is a novel CYPSec inter-sensor communication scheme for BANs, which utilizes specific physiological signals as a way of enabling sensors to agree on a common key, thereby forming a basis for maintaining communication security. Physiological signals are stimuli generated from the various functions performed by the human body. Examples of physiological signals include heart-rate, temperature, and blood glucose level. The idea of using physiological signals for securing inter-sensor communication was first introduced by us in [23] and studied in detail in [130] [131].

The PSKA scheme is designed for establishing secure unicast channel between sensors in a BAN, and principally works in four phases: 1) The two communicating sensors measure a pre-agreed physiological signal, 2) The sampled signal is processed and features are extracted from it, which are then quantized into a binary string; 3) One of the two nodes (designated sender) then uses an arbitrary key to secure (encrypt/integrity protect) any data it wants to communicate; 4) It then hides the key using this binary representation of the physiological signal feature and transmits both the hidden key and secured data as a single message; 5) The receiving node retrieves the arbitrary key using the local version of physiological signal and checks it by verifying (decrypting/checking integrity) the data received. Once these steps are executed, any subsequent secure communication between the sensors can take place without explicit physiological signal measurement using the key that they just exchanged (except in special cases, for example if the network is being re-configured). The scheme achieves its security based on the observation that the physiological signal features generated from a subject are unique at that time. An adversary who is not in contact with the host's body will not be able to accurately measure the physiological signal and therefore cannot influence the secure communication taking place between the sensors. We shall formalize and demonstrate these properties of PSKA further into the discussion.

3.5.1. Physiological Signals: Issues and Properties

In order to implement the PSKA protocol, we need to understand in some detail the principal issues pertaining to utilizing physiological signals for enabling secure communication in BANs. The first question is of course which physiological signal to use. The list of physiological signals generated by the human body is enormous, defining the criteria for choosing physiological signal is therefore very important. Once the signal has been chosen, they have to be measured - sampled - at both the sensors. Many issues exist in this regard including how to synchronize the measurement, how maintain synchronization between the nodes, how to process the data collected and how to reconcile between the inherent difference in the signals when measured at different parts of the body. Once the measurement is completed, the signals are processed to generate features which are then used for key agreement process. We describe each of these issues in detail below.

3.5.1.1. Choice and Measurement

The human body produces many physiological signals as a result of its “operation”. Physiological signals, to be useful for secure key agreement, need to have the following properties or **design goals**, namely - 1) *Length and Randomness*: The keys agreed upon are long and random to prevent brute-forcing; 2) *Low Latency*: The duration of physiological signal capture required is minimal; 3) *Distinctiveness*: Knowing the feature derived from the current value of the physiological signal of one subject will not provide significant advantage in guessing the keys being agreed by sensors on another host. An important characteristic of distinctiveness is that it authenticates the communicating sensors by ensuring that only sensors on the same BAN can agree on a shared key; and 4)*Temporal Variance*: Knowing the physiological signals at any time will not provide significant advantage in knowing the keys agreed upon in future executions of the scheme.

At this point, we would like to *differentiate the proposed technique from biometrics systems* such as [87] [95] [122]. PSKA is fundamentally different from traditional biometric systems because the physiological signal based features used by it show both distinctiveness and temporal variance properties (see Section 3.7.3). Biometric systems only posses the former, and actually depend on not possessing the latter. A biometric template created using a person’s fingerprint, for example, is used for all subsequent authentica-

TABLE 1

Range of Common Physiological Signals, albeit unusable for PSKA

Physiological Values	Range
Blood Glucose	64-140 mg/dL(varies with activity)
Blood Pressure	120-160 mmHg (systolic)(Range is from hypo-tension to hypertension)
Temperature	97.0-105.0 F (Range across ages and normal and abnormal conditions
Hemoglobin	12.1-17.2 g/dL(Varies between male female and age and altitude
Blood Flow	$>$ 0.9 ABI(normal), $<$ 0.5 ABI (abnormal)

tion attempts made by the subject. The success of the authentication is based on the closeness of the match between fingerprint sample, presented by the subject, and the original template. This model of operation allows arbitrary time to pass between two biometric measurements, as the biometric template itself is invariant. However, in the case of PSKA physiological signal features are time variant. Key un-hiding process at the receiver therefore requires a (loosely) synchronized measurement of physiological signals with the sender, in order to be successful. An attacker knowing the physiological signal based features at any given time, has no advantage in compromising any past or future key agreements between sensors, a property that biometric templates do not posses. Identifying physiological signals which possess the properties specified above is an important step in implementing PSKA. As we shall see in Section 3.7 that Photoplethysmogram (PPG) and electrocardiogram (EKG) signals processed in a particular manner (in frequency domain) provide these properties.

3.5.1.2. Topographic Specificity

Another important issue to be considered with physiological signals is the topographic specificity of the human body. That is, a given physiological signals measured from different areas of the body appear to have similar trends but not the exact same values when measured at different parts of the body [77]. As a result, the information symbols in the keys get re-ordered leading to translational and rotational errors [54] which produce drastically different values. To overcome this problem of differences in the value of physiological signals, we proposed in our preliminary work [23] the treatment of the differences as analogous to the introduction of error while transmitting data between communicating sensors. The choice of error correction mechanism is an important issue that needs to be considered carefully.

3.5.1.3. Physiological Signal Processing

One of the important properties of the physiological signals is that they are time-variant and vary unpredictably. This prevents malicious entities from guessing the current value of the physiological signals or knowing future values if the current value is known. But this temporal variation also poses problems for the sensors trying to use the physiological signal to communicate securely. We therefore need to ensure that the communicating sensors see identical (baring the topographic specificity) copies of physiological signals. A very important question in this regard is: what kind of synchronization is required for PSKA? This will of course depend upon how the physiological signal are processed.

Signal processing of physiological signals can be done in two ways: through a *time-domain* or *frequency domain* analysis. Time domain analysis signal values are directly quantized into binary streams and concatenated as keys. However, due to the time variance of physiological signals, the time-domain analysis tends to generate different features if the communicating sensors are not synchronized. To estimate the synchronization required for time-domain analysis we downloaded EKG and PPG data from the MIT PhysioBank MIMIC database (http://www.physionet.org/physiobank/database/mimicdb/). For the eleven patients whom we considered (based on the availability and completeness of the EKG and PPG data), the average time difference between a EKG peak and the corresponding PPG peak was found to be μ = 250 msec, with standard deviation of $\sigma = \pm$ 6 msec. Therefore we posit that to utilize IPI for securing inter-sensor communication, the level of synchronization required should be in hundreds of milliseconds. Given the capability of sensor synchronization protocols [38] to reach a few microseconds of synchronization, we contend that for measuring IPI we need only a relatively loose synchronization between the communicating sensors. Note that different physiological signals will have different synchronization requirements. If EKG were to be directly used then the level of synchronization would be limited by the sample rate (f). The data we utilized was sampled at 125Hz i.e. one sample was generated every 8 msec. To ensure no data points are missed at either sensors, the sensor clocks have to be synchronized to within 8 msec. A frequency domain analysis, on the other hand tends to produce similar values even if the synchronization between the sensors have very loose synchronization. Our experiments (see Section 3.7.3 show that even 1 second difference in measurement start times does not have any adverse affects on PSKA for PPG and EKG signals. However converting the time-domain signal into frequency domain requires mathematical transforms which can be expensive.

3.5.1.4. Physiological Signal Measurement

Now that we know that the level of synchronization required for PSKA to function we have to determine when two nodes have to start measuring physiological signals. With loose synchronization required, this can be achieved in a simple manner:

- When a node (designated sender) wants to communicate with another, it sends a *measurement-sync* (with the ID of the receiver in it) message to the receiver and starts measuring the pre-chosen physiological signal.
- The propagation delay is considered to be effectively zero, as the signal (electromagnetic radiation) travels at the speed close to c (speed of light in free space), and the synch message has to typically reach the distance of about 0.5m on an average. The few nanoseconds that the signal will take to cover the distance is well within the margin of error that can be tolerated by our synchronization scheme.
- The receiver, on receiving the message starts measuring the physiological signal.

The act of broadcasting the *measurement-sync* message in the open does not cause any security threats because the only consequence of sending a *measurement-sync* message is physiological signal measurement. No valid message exchange can take place as a result of the synchronization because the adversary has no way of generating a legitimate physiological signal based key, and re-using old physiological signal measurements will also not work because of the time-variance of physiological signals. Further, as PSKA key agreement process is sufficiently strong (about 95 bits, see Sections 3.6 and 3.7.3), the broadcast of *measurement-sync* message will be rare, as a single measurement of physiological signal can be used for some time (about 135 years [108]). Further, in order to minimize the chances of a malicious entity broadcasting continuous

```
PSKA_SEND()
if(Data to Send)
    if(PVExisits == false)
        send(measurement-synch)
        Measure chosen PS ($PS_s$)
    endif
    Generate RandKey
    $\gamma$ = Hide(RandKey, $PS_s$)
    mac = MAC(RandKey, $\gamma$, Nonce)
    msg = <mac, $\gamma$>
    send(msg)
endif
```

```
PSKA_RECV()
Packet = receive()
if(Packet == measurement-synch)
    Measure chosen PS ($PS_r$)
if(Packet == msg)
    RandKey'' = Unhide($PS_r$, $\gamma$)
    mac' = MAC(RandKey'', $\gamma$, Nonce)
    if(mac == mac')
        Accept RandKey''
        send(ACKN)
    else
        Reject RandKey''
endif
```

Fig. 6. PSKA Execution Conceptual Overview

measurement-sync message to deplete the node's battery, it can be mandated that the synchronization be carried out in a safe location (doctor's office) where the process can be executed with relative safety.

It is possible that between the times the *measurement-sync* message is received and the physiological signal measurement begins, the clock (at the receiver) starts to drift. This drift, for a typical crystal oscillator, is usually of the order of 100μsec per second and will not affect the physiological signal synchronization [38]. It is also possible that the sensors receiving the *measurement-sync* message do not start at exactly the same time due to reception delays. In [38] the maximum reception delay in the case of Berkeley motes has been reported to be about 53μsec. Assuming a similar value for BSNs, the misalignment in the start of the physiological signal measurement should not be significant. Similarly, platform specific delay factors such as interrupt requests and sleep cycles will also not be a cause for concern.

3.5.1.5. PSKA: Conceptual View

Given all the issues involved in using physiological signals, before we go onto with details of PSKA works, we now present a conceptual overview of the PSKA protocol for securing inter-sensor communication. Figure 6 shows the working of the PSKA protocol for secure (integrity protected) unicast communication. Assuming both the sensors have never communicated before (determined by $KeyExisits$ flag), both the sender and the receiver synchronously measure a pre-agreed physiological signal. Now, if the sender has data to send it will: 1) Generate an arbitrary key called $RandKey$; 2) Hide the $RandKey$ using its version

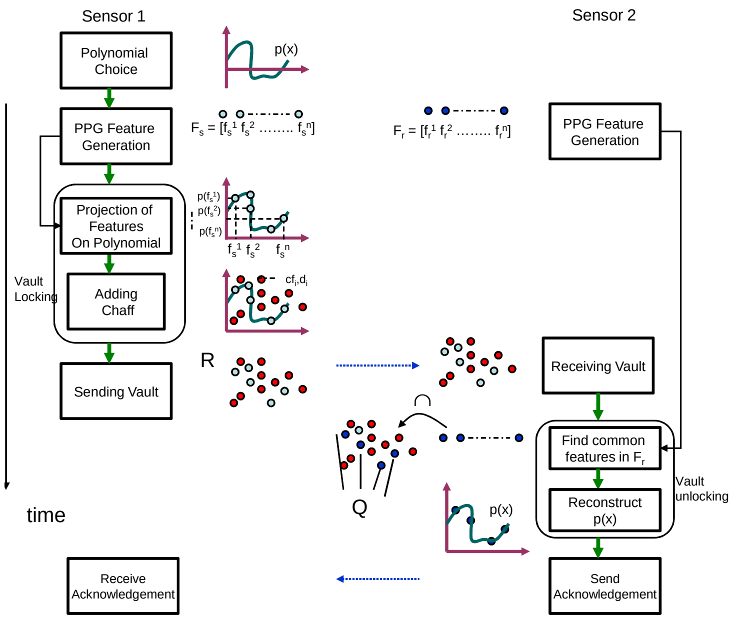


Fig. 7. Key Agreement in a BAN using PSKA

of physiological signal (PS_s) to form γ, and concatenate a Message Authentication Code (MAC) on γ, computed using the $RandKey$; and 3) Transmit the hidden key and MAC to the receiver. The receiver on receiving this message: 1) Retrieves the key from γ using the local version of physiological signal (PS_r) and an error correction function; 2) The resulting $RandKey''$ is used to compute the MAC on γ received and compared with the mac received, if they match the $RandKey''$ is accepted else it is rejected. The successful verification of MAC, assures the receiver that the sender is a sensor on the same host (BAN) as itself. Once the key has been agreed upon, it can be used for securing future communications between the two nodes. The PSKA key agreement process meets all the security requirements of PHMS - integrity (using the MAC), confidentiality (using the Hiding process), authentication (successful verification of the MAC), and usability (key agreement based on physiological signals). A key question in this regard is how the hiding and un-hiding process takes place. To answer this question we now move on to the specifics of our solution.

3.5.2. PSKA Protocol Execution

The purpose of *Physiological Signal based Key Agreement (PSKA)* is to *facilitate secure inter-sensor communications* between two sensors by enabling them to agree upon a pair-wise symmetric key, using physiological signal based features. A major challenge in implementing error correction approach described above is identifying the error correction technique which can be used to correct the difference in physiological signals. The error correction functions should not be able to correct the presence of a few differences in feature vectors, they cannot handle re-ordering of, or presence of additional features (in one of the sensors) in the feature vector [54]. One of the ways of addressing this problem is to use of a cryptographic construct called *Fuzzy Vault* [54]. The fuzzy vault construct is the physiological certificate

3.5.2.1. Fuzzy Vault

The fuzzy vault scheme first proposed in [54] is designed to lock (hide) a secret (S) in a construct called a *vault* using a set of values A. Once the vault has been locked, it can be unlocked only with another set of values B which has a *significant* number of values in common with set A. The construction and locking of the vault is done by: 1) generating a v^{th} order polynomial p over the variable x that encodes the secret S, 2) computing the value of the polynomial at different values of x from set A and creating a set $R = \{a_i, p(a_i)\}$, where $1 \leq i \leq |A|$, and 3) adding randomly generated set of points called *chaff* to R which do not lie on the polynomial. Once the vault is constructed, unlocking it based on the set B is done by constructing a set $Q = \{(u, v)|(u, v) \in R, u \in B\}$. The unlock process is possible only if Q has a significant number of legitimate (non chaff) points which are on the polynomial [54]. We can map this scheme onto PSKA by setting the physiological signal features obtained at one of the sensors to set A, those obtained at the other to set B, and generating a polynomial whose coefficients form the secret key that needs to be agreed upon.

Consider the following example which illustrates the operation of the fuzzy vault. Let the polynomial be $p(x) = x + 1$, set $A = \{1, 2, 3\}$ and $B = \{1, 3, 4\}$, then the vault R created by computing the polynomial's value at each point in A is $R = \{(1, 2)(2, 3)(3, 4)(4, 7)(6, 9)(7, 12)(8, 5)\}$. The last four points are the chaff points which do not fall on the polynomial. To unlock the vault the set Q is constructed, where

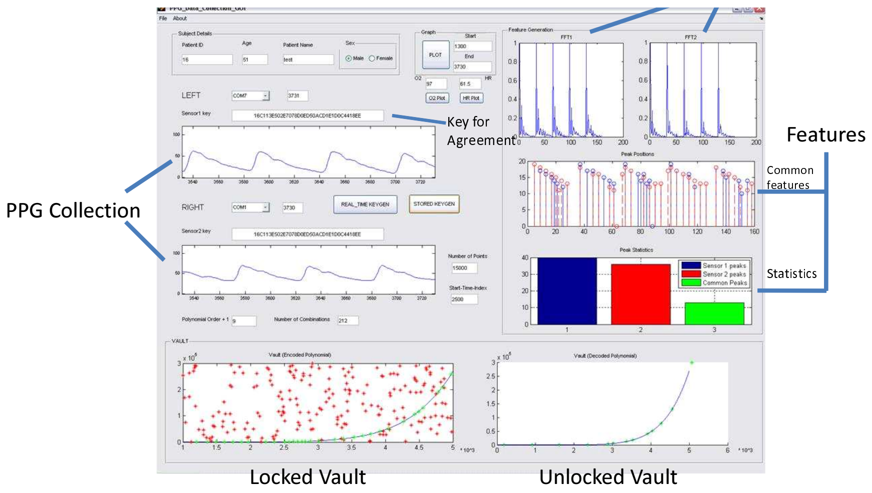


Fig. 8. Screen Shot of execution of PPG based PSKA for data collected from same person

$Q = \{(1,2)(3,4)(4,7)\}$. As the set Q has two points on the polynomial, we can use it to easily reconstruct the first order polynomial and thus unlock the secret.

3.5.2.2. Vault Locking & Unlocking

The use of polynomials ensures that the sets A and B need not have any order to them as long as they have a significant number of common values. The presence of the chaff points adds security to the vault, hides legitimate points and the actual polynomial. Unless the adversary knows a large number of points on the polynomial, it cannot reconstruct the polynomial. In this section, we show how fuzzy vault scheme for key agreement with PSKA.

We use the term *sender* for the sensor which creates the vault and locks it, and the *receiver* for the sensor that unlocks the vault to access the secret key. The key agreement occurs as follows:

Feature Generation: First, both the sender and the receiver obtain physiological signal based features. This is a four step process: 1) Both the sensors sample the physiological signal in a loosely synchronized manner, at a specific sampling rate for a fixed duration; 2) The samples are divided into windows and a Fast Fourier Transform (FFT) is then performed on each of these parts; 3) The FFT coefficients of each of the overlapping windows (a pre-defined number of contiguous signal time-series points) are then passed through a peak detection function (a simple local maxima detector) which returns a tuple of the form $< k_x^i, k_y^i >$, where

k_x^i is the FFT point at which peak is observed (its the peak location on x-axis, also called *peak-index*), k_y^i is its corresponding FFT coefficient values (magnitude of the peak, or *peak-value*), and i is the index of the peaks. The number of peaks observed by a sensor vary upon situation; 4) Each of these peak-index (k_x) and peak-value (k_y) pairs are quantized and converted into a binary string and concatenated ($[k_x|k_y]$) to form a *feature*. Individual features obtained from a single measurement are grouped together to form a feature vector $F_D = \{f_D^1, f_D^2, ...f_D^N\}$, where f_D^l = $[k_x^l|k_y^l]$, D is either the sender (s) or receiver (r) node, and N is the size of the feature vector, i.e., number of indexes where peaks were observed. The values of the different parameters used for feature generation are dependent upon physiological signal used and they need to be tuned during deployment. We chose FFT peaks as features for two reasons -1) they are simple to detect, and 2) they characterize a subject's physiology very well. They are ideal for distinguishing between sensors which are in the same BAN or different BANs, thus providing an efficient authentication mechanism and a basis for key agreement (see Section 3.7 for more details). At the end of the feature generation process, the sender and receiver possess feature vectors of the form $F_s = f_s^1, f_s^2, ...f_s^N$ and $F_r = f_r^1, f_r^2, ...f_r^N$, respectively.

Polynomial Choice: Once the features are generated, the sender generates a v^{th} order polynomial of the form $p(x) = c_v x^v + c_{v-1} x^{v-1} + ... + c_0$, where the value of the coefficients (c_is) are selected randomly (using a pseudo random number generator, for example). The order of the polynomial (v) used within the BAN is not a secret and known to all sensors in the network. The coefficients, concatenated together, form the secret key that the sender wants to communicate to the receiver ($Key = c_v|c_{v-1}|...|c_0$). We have set the length of this Key to be 128 bits (longer keys can easily be used), and depending upon the order of the polynomial used, the coefficients are obtained by dividing the Key accordingly.

Vault Creation: With the polynomial and feature vector available, the sender now creates the fuzzy vault, by computing the set $P = \{f_s^i, p(f_s^i)\}$, where $f_s^i \in F_s$, and $1 \leq i \leq N$. It also computes a much larger set of M random chaff points of the form $C = \{cf_j, d_j\}$, where $cf_j \notin F_s$, $d_j \neq p(cf_j)$, and $1 \leq j \leq M$. Each chaff point, cf_i, is within the same range (0-2^{13}) as that of the features. Therefore, 2^{13} is the bound for the total number of points in the vault ($|R|$), which is equal to $|M| + |N|$. We refer to the cardinality of the set R as the *vault size*.

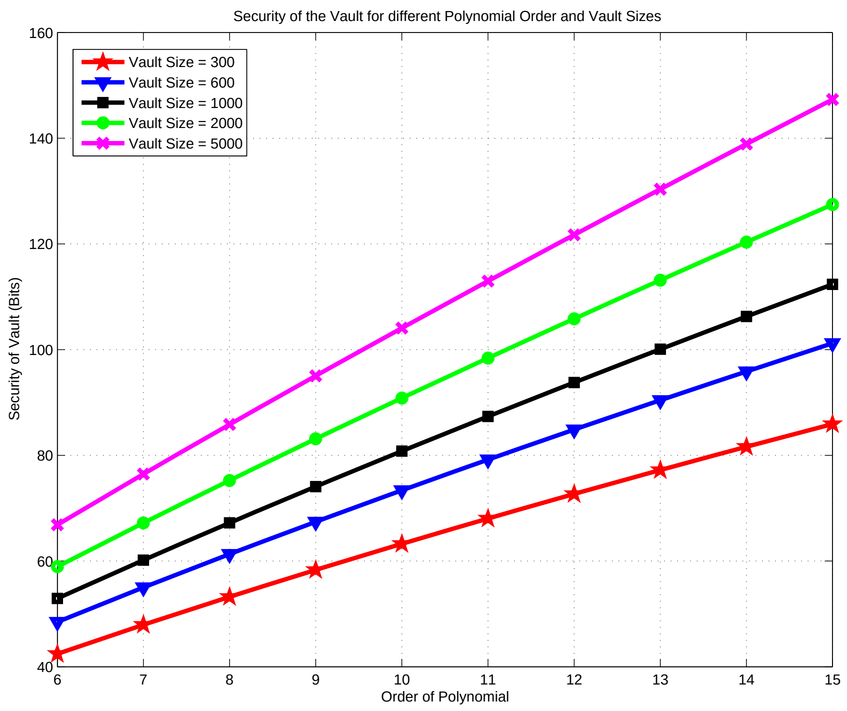
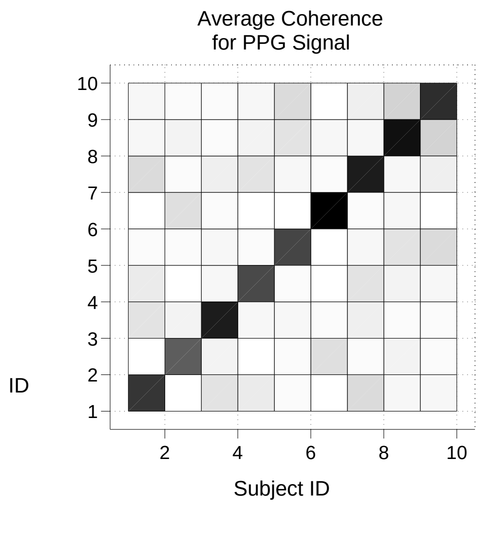


Fig. 9. Vault Strength w.r.t. Polynomial Orders for Different Vault Sizes

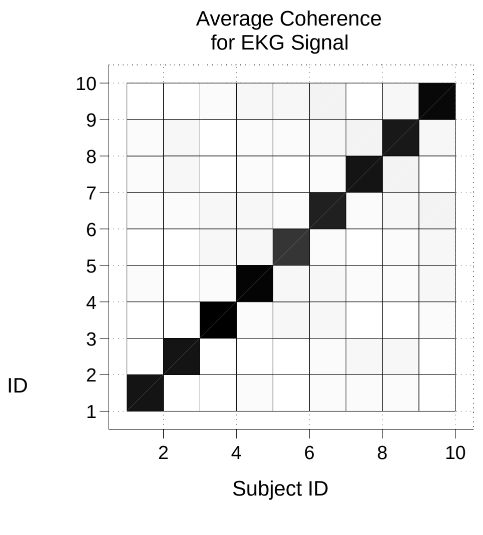
Vault Locking: The sender then randomly permutes the values in the vault $R = RandPermute(P \cup C)$, to ensure that the chaff points and the legitimate points are indistinguishable. The cardinality of the set C can vary with respect to the level of security needed. The larger the set C, the more difficult it is to break the vault. Section 3.7 discusses the relationship between the vault's size and its security in more detail. This set R is essentially γ from the our conceptual overview.

Vault Exchange: The sender communicates the vault R to the receiver using the following message: $Sender \rightarrow Receiver : IDs, IDr, R, No, MAC(Key, R|No|IDs)$. Here, IDs and IDr are the ids of the sender and receiver, respectively, No is a nonce (unique random number) for transaction freshness; MAC is a message authentication code (e.g. HMAC-SHA1); and, the key (Key) used is the one that is being locked in the vault. This message is essentially $CERT$ from the our conceptual overview.

Vault Unlocking: The receiver upon receiving the vault R first computes the set Q, where $Q = \{(b, c)|(b, c) \in R, b \in F_r\}$. It then tries to reconstruct the polynomial p based on the points in Q us-



(a)



(b)

Fig. 10. Average Coherence computed over all time stamps for all subjects a) PPG , b) EKG

ing the Lagrangian interpolation (as suggested in [122]), according to which, the knowledge of $v + 1$ points $\{(x_0, y_0)(x_1, y_1),...,(x_v, y_v)\}$ on a polynomial allows the reconstruction of a v^{th} order polynomial by performing the following linear combination: $p'(x) = \sum_{j=0}^{v} y_j d_j(x)$, where $d_j(x) = \prod_{i \neq j, i=0}^{i=v} (x - x_i)/(x_j - x_i)$. For the receiver to be successfully able to unlock the vault, the condition $|Q| > v$ should hold. It then takes $v + 1$ points (from Q) at a time and tries to unlock the vault. The coefficients of the resulting polynomial are then used to verify the MAC. This not only confirms the correctness of the unlocking process, but also authenticates the sender to the receiver (confirms that the sender is on the same BAN as the receiver). This is because of the distinctiveness and temporal variance property of the physiological signal features which ensures: 1) the features generated from physiological signals for PSKA are drastically different for two different people, and 2) the old vaults cannot be replayed as the features would have changed by that time, and cannot be unlocked (see Section 3.7 for more details).

Vault Acknowledgment: If unlocking was successful, the receiver then sends a reply back to the sender to inform it of its correct unlocking of the vault using the following message: $Receiver \rightarrow Sender : MAC(Key, No|IDs|IDr)$. The symbols have the same meaning as above. The successful verification of the acknowl-

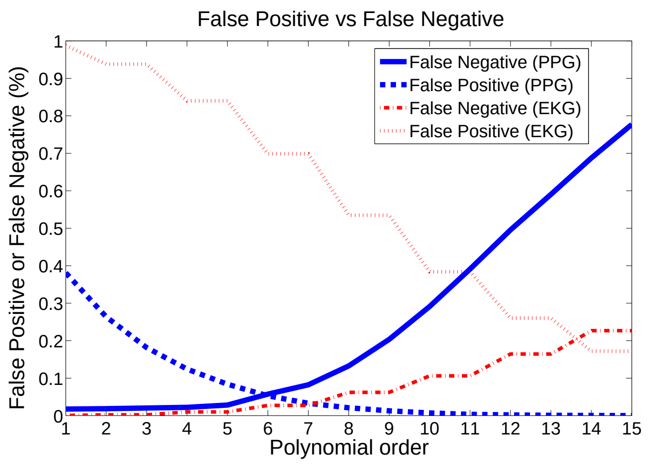
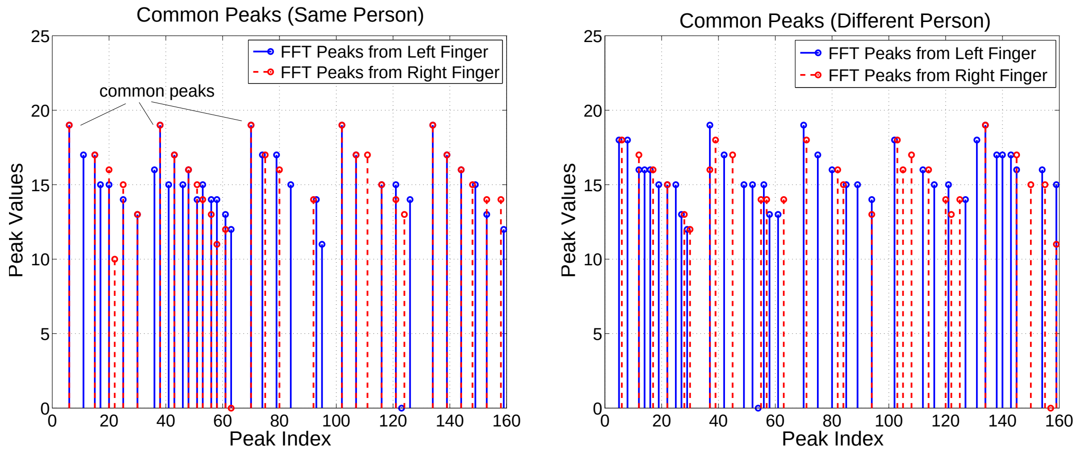


Fig. 11. Benchmark Implementation - False Positive vs. False Negative Rate for EKG and PPG

edgment authenticates the receiver to the sender. This is because only a node on the BAN (receiver), which measured the same physiological signal at the same time as itself, could have unlocked the vault given the distinctiveness and temporal variance properties of the physiological signal based features. Section 3.7 provides more details on how these two properties are met.

3.5.2.3. Discussion

Figure 7 shows the feature generation process. We refer to the execution of these seven steps as an *iteration* of PSKA. The random key (Key) generated in the first step can be used to enable confidential, authenticated and integrity protected communication between sensors in a *plug-n-play manner making BANs more usable*. None of the traditional key distribution schemes [39] [74] [142] nor physiological signal based approaches [93] consider this property. Further, with PSKA, no random key or physiological features are ever re-used. This ensures any knowledge of past keys or physiological features (due to their temporal variance property) of a host cannot be used for subverting the vault. Once the PSKA is successfully executed, both the sender and receiver have the common random Key. In order to use this key for future data communication we derive new data communication keys, as suggested in as in [90], one at the sender $K's = f(Key, 1)$ and another at the receiver $K'r = f(Key, 2)$ to minimize the effects of cryptographic attacks on secured data communication, here f is a pesudorandom function.



(a) (b)

Fig. 12. Benchmark Implementation - PPG FFT Peaks (peak index vs. peak values) for: a) Same host (Total Common Peaks: 12), b) Different hosts (Total Common Peaks: 2)

The one-hop security provided by physiological signals can be easily extended to multi-hop end-to-end communication, where a physiological signal based key is generated between each link on the path to the base station. This might increase the latency within the network which might be a problem if BAN is being used in emergency situations. To minimize this, sensors can execute PSKA once with their neighbors, arrive at a common key, derive keys from this common key, and use for more than one communication. As more and more medical sensors/devices with wireless communication capability are being developed, interoperability (plug-n-play interaction) among these entities is becoming a growing issue [135]. PSKA removes the hurdles in the process from a security stand-point, by enabling sensors to establish a secure communication channel without any manual intervention. However, larger issues for interoperable medical sensor systems such as communication technology, data and command format are still open problems.

3.6. Security of PSKA

Security issues in PSKA primarily arise due to its vault exchange requirement and reply it to authenticate oneself and join the BAN. An eavesdropper can record the vault and try to construct the hidden polynomial (key) from it. In this section we discuss the security implications of the two principal aspects of PSKA: the vault and its exchange.

3.6.1. Vault Security

The use of the fuzzy vault construct in PSKA ensures that, even though the two sensors may not have all the features in common, they can still agree upon a common key in a secure manner. The security of the PSKA scheme is based on the difficulty of polynomial reconstruction. The hiding of the legitimate feature points among much larger number of the bogus chaff points, whose values are in the same range, makes the job of identifying the legitimate points very difficult. An adversary who does not know any legitimate points (as it cannot measure the relevant physiological signals from the host's body) has to try out each of the $v + 1$ points in set R to be able to arrive at the correct polynomial. By the same account, the more the number of features an entity is aware of, the easier it is to re-construct the hidden polynomial. An example is the receiver who has to try out $v + 1$ points from the set Q to arrive at the correct polynomial [3].

Figure 9 shows the strength of the vault for different values of polynomial order used for different number of chaff points. The strength of the vault is determined by the number of combinations an adversary has to try in order to find $v + 1$ legitimate points in the vault. For ease of understanding, we represent this computation requirement in terms of its equivalence to brute-forcing a key of a particular length (bits). As expected, increasing the number of chaff points increases the security provided by the vault. Higher the order of the polynomial, the more common features we need to find and therefore higher the security. Note that PSKA guarantees successful unlocking of the vault as long as the number of common features in Q are greater than v. By choosing the order of the polynomial to a value $|F_s \cap F_r'| < v < |F_s \cap F_r|$, where $|F_s \cap F_r'|$ are the number of common features between feature vectors of two different individuals and $|F_s \cap F_r|$ are the number of common features between feature vectors for the same individual, we can ensure successful vault unlocking for the receiver but not for the adversary. Of course, this will work only if there is a discernible

[3]It needs to be pointed out that recent years have seen the development of remote physiological monitoring technologies [34] [43]. Such technologies in their current state are designed to work in a controlled environment and are susceptible to interferences by objects in the environment, directionality, and motion artifacts. The use of such a technology to measure the physiological signals in open environments and compromise PSKA is an open issue and merits further investigation.

TABLE 2

PSKA Feature Generation Parameters

Signal	**Parameters**	**Values**
PPG	Sampling	60 Hz
	Sampling Duration	12.8 secs
	FFT	256 points, 5 windows[a]
EKG	Sampling	125 Hz
	Sampling Duration	4 secs
	FFT	256 points, 2 windows[b]
PPG/EKG	Peak Value Quantization	5 bits
	Peak Index Quantization	8 bits
	Feature Length	13 bits

[a] First 32 points per window were concatenated together for peak based feature generation
[b] First 128 points per window were concatenated together for peak based feature generation

difference in the number of features between physiological signals collected from the same individual and different individuals. More on this in Section 3.7.

From Figure 9, it can be seen that choosing vault parameters is a balancing act between multiple criteria: number of chaff points, feature length and number of common features. We recommend that the number of chaff points should be as large as possible compared to the total feature length as can be tolerated by the system. A vault with an overwhelming number of chaff points is difficult to compromise by pure brute-force. This along with an appropriate polynomial order depending upon the common feature statistics will ensure that vault is secure.

3.6.2. Exchange Security

The vault exchange and acknowledgment phases make it very difficult for adversaries to know the key being agreed upon, because of the following reasons:

- The presence of IDr in the vault exchange message tells the sensors in the vicinity of the sender, who the intended receiver is. The nonce No is used to maintain the freshness of the protocol, i.e. to ensure that the acknowledgement received is in response to its latest transmission.

- If a malicious entity sends a vault exchange message (by replaying previous exchanges or creating its own vault using old physiological features) it will be discarded by any receiver as the MAC would not match due to difference in the Key used given the temporal variation of the physiological features.

- The vault has a many orders of magnitude more number of chaff points compared to the legitimate points (for example, 1000 chaff points to 30 legitimate feature points) which makes it difficult for adversaries to know which points are legitimate and which are not (as discussed above). A malicious entity which cannot unlock the vault cannot send a valid acknowledgement as it would have to generate a valid MAC without the Key.

- A malicious entity trying to mount a man-in-the-middle attack has to be aware of the physiological signal features being used. Without them, any modification of the vault during exchange would be caught as none of the $\binom{|Q|}{v+1}$ keys unlocked by the receiver will verify the MAC.

Thus we can see that PSKA meets all the security requirements enumerated earlier, it keeps the key agreement confidential (using the vault), integrity protected (with the MAC), authenticated (the successful unlocking of the vault which shows the receiver is on the same BAN (more details in next section)) and of course usable, as no manual key pre-deployment is required.

3.7. PSKA Benchmark Implementation

In this section we validate our approach by describing a proof-of-concept implementation of PSKA using Matlab. We use this implementation as benchmark for later actual hardware based implementation. Figure 8 shows the screen shot of our implementation of PPG-based PSKA in Matlab which can perform key agreement based on data collected in real-time from a host.

3.7.1. Physiological Signal Choice

For our benchmark implementation, we validated the PSKA approach using two of the most common physiological signals that can be collected from a person - photoplethysmogram (PPG) and electrocardiogram (EKG). The former is a measure of the volumetric change in the distension of arteries due to the perfusion of blood through them during a cardiac cycle, while the latter is the representation of a host's cardiac cycle generated by the electrical activity of the heart. The PPG data utilized for our analysis was collected from 10 volunteer subjects in the IMPACT Lab. We used Smith Medical pulse oximeter boards (specifications can be found at http://www.smithsoem.com/applications/oxiboards.htm) to collect the data from the volunteers. The

TABLE 3

Benchmark Implementation: PPG/EKG Feature Statistics

Signal	**Parameters**	**Values**
PPG	Number of Iterations	113, 1.6 sec apart
	Avg. Feature Vector Length	$\sim$30
	Avg. # Common Features (Same Host)	12
	Mode # of Features (Same Host)	14.8
	Avg. # Common Features (Different Hosts)	2
	Mode # of Features (Different Hosts)	0.8
EKG	Number of Iterations	180, 4 sec apart
	Avg. Feature Vector Length	$\sim$87
	Avg. # Common Features (Same Host)	24.7
	Mode # of Features (Same Host)	25.2
	Avg. # Common Features (Different Hosts)	9.5
	Mode # of Features (Different Hosts)	8.1

volunteers were asked to sit upright with their hand firmly placed on a desk; an oximeter sensor was placed on the index finger of each hand. We assume for the purposes of our experiment that the two communicating sensors utilize signals from the sensor on each finger. Data was collected for about 5 minutes from each host at a sampling rate of 60Hz. The EKG data - for 10 subjects - on the other hand was obtained from the PhysioBank database (http://www.physionet.org/physiobank). EKG data was obtained from two leads on every person. We assume that the two communicating sensors utilize signals from each lead for key agreement. About 15 minutes of data was downloaded for our analysis, with the signals sampled at 125Hz.

3.7.2. Suitability of Chosen Physiological Signals

PSKA depends upon the high degree of frequency-domain correlation between loosely synchronized physiological signals (EKG/PPG) of the same host to enable key agreement between sensors deployed on a host. In order to determine if the chosen physiological signals can be used for our purposes, we performed a rigorous study on the frequency domain correlation of EKG and PPG signals. The metric we used for this purpose was the *coherence* of time synchronized EKG and PPG signals. Coherence is a measure of the cross correlation of FFT coefficients of two signals. Formally, coherence of two time series signals x and y can be calculated as $\frac{Cxy^2}{(Cxx \times Cyy)}$, where Cxy is the cross-correlation of FFT coefficients of x and y, Cxx is the autocorrelation of the FFT coefficients of the signal x and Cyy is the autocorrelation of the FFT coefficients of signal y. We used coherence because it provides us with a measure of commonality in the frequency domain characteristics of the two EKG and PPG signals.

In order to compute the coherence, we divided our PPG and EKG time-series into slots which were 1.6 seconds apart. Each slot was long enough to contain 256 sample points (for PPG with 60Hz sampling frequency, it was about 4 seconds and EKG with 125 Hz sampling frequency it was 2 seconds). We choose 256 sample points because the PSKA uses a 256 point FFT. We computed the coherence between the PPG and EKG signals for all combination of subjects, and derived a metric from it to represent the overall trend called *average coherence*. Average coherence is the normalized area under the coherence curve of two EKG and PPG signals, averaged over all possible start-times of the slots(over 100 start-times for both EKG and PPG). Figure 10 (a) and (b) shows the average coherence results for PPG and EKG signals for all possible combinations of subjects. It can be seen that, for the same host (diagonal values), the metric has a very high value indicating that a loosely time synchronized measurements of EKG and PPG signal has high frequency domain commonality. However, for different host the metric has very low values indicating low commonality. The results thus show that frequency domain components of PPG and EKG signals are highly correlated and ideal for executing PSKA.

3.7.3. Performance Analysis

The Table 2 shows the parameters used in the feature generation process. The basis for validation was the meeting of our design goals. Our aim is to two fold - 1) show that the results follow the design goals set forth earlier, and 2) perform a comparative analysis of PPG-based and EKG-based PSKA.

Long and Random Keys: The keys to be agreed upon are generated by the sender in the form of polynomial coefficients using a pseudorandom number generator. The length and randomness of the keys agreed can therefore be ensured.

Low Latency: The duration of sampling needed for secure key agreement varies with physiological signals, and the technique used. With PPG which is sampled that 60Hz, our best results were obtained with 12.8 seconds of data (due to the multiple windowed FFT used to generate features), with EKG which was sampled

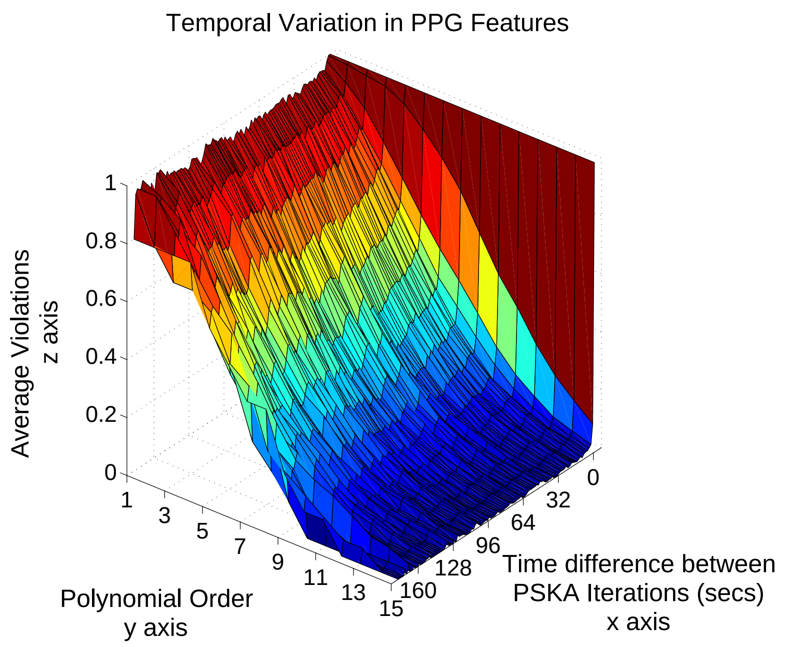


Fig. 13. Benchmark Implementation Temporal Variance in PPG Features

at 125Hz, this dropped down to 4 seconds of data. In general we observe that - the more detailed the data available, lower the latency.

Distinctiveness: An important requirement of PSKA is that the physiological signals can distinguish people. This is important because we do not want the vault created by a sensor in one BAN to be unlocked by another sensor located on another subject (either accidentally or maliciously) based on features generated from its measurements. We therefore want to make sure the number of common features for sensors on the same subject be "significantly" more than the number of common features for sensors on the different subject. Our definition of significant is dependent upon the polynomial order v used. Figure 12 shows peak values versus peak index graph for PPG and EKG obtained from the same individual and different individuals. The lines which are completely super-imposed are the common features. We found that the higher the correlation between the FFT of PPG and EKG signals at two sensors, the larger is the number of peaks values and indices they have in common in their respective FFTs. It can be seen from Figure 12, sensors on the same person have a higher number of super-imposed lines (less blue bars are visible as they are hidden by the red ones) compared to sensors on two different people. The lack of correlation between the FFTs of the physiological

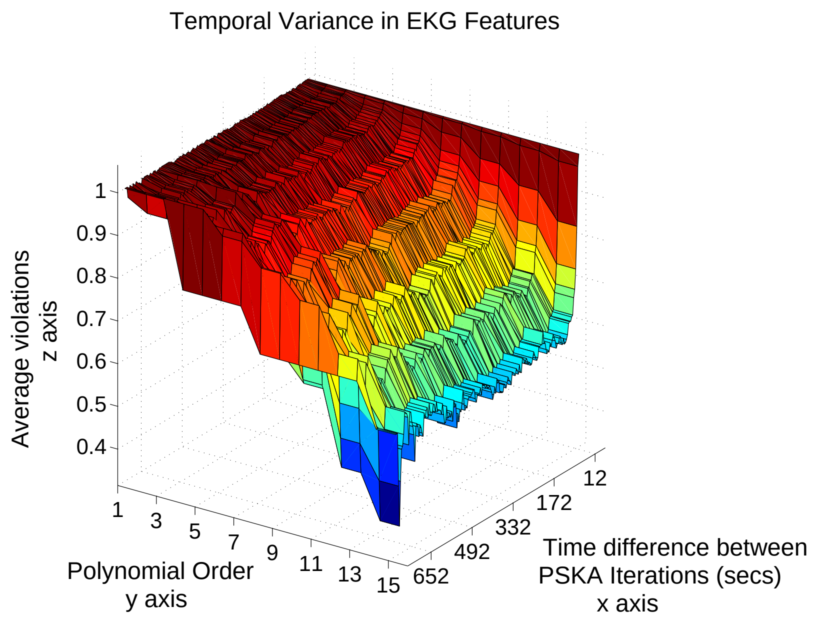


Fig. 14. Benchmark Implementation Temporal Variance in EKG Features

signals measured at two different subjects is due to the difference in physiological signature of each person at any given time.

Table 3 shows the statistics observed for the features when PSKA was executed based on PPG and EKG signals. We can see that, the difference between the number of common features between two sensors on the same subject and two sensors on two different subjects, is significant. Therefore, given the statistic on differences in the number of common features, we can now decide the possible values for v. The polynomial order has to be such that we minimize the *false positives*, i.e. the number of times the common features between two people exceeds it, as well as minimize the *false negatives*, i.e. the number of times the common features for the same subject is below it. Figure 11 shows the percentage of false positives and false negatives when PPG and EKG is used with PSKA for different orders of polynomials. For PPG, the false positive and the false negative rates are minimized when the order of polynomial used is 6, while for EKG it is 14. From these results we can say that using features derived from the PPG and EKG signal to generate a vault, does not give any significant advantage to an adversary who uses features derived from another subject, provided an appropriate polynomial order is chosen.

Temporal Variance: We have shown EKG and PPG signal features are distinctive between two people. We now want to establish their temporal variance. Figures 13 and 14 show the temporal variance of the PPG and the EKG signal features, respectively. The x-axis of the graph is the time difference between the PPG and EKG measurement start-times of two iterations of PSKA, the y-axis is the polynomial order used, while the z-axis shows the *average violations*, which is the percentage of times when the number of common features between the first and second iterations of PSKA are greater than the order of the polynomial used[4]. As expected, when the time difference between the two iterations of PSKA for both PPG and EKG is very close violations are very high; since the feature values in both the iterations are very similar. However, as the time difference increases the percentage of violations falls drastically for PPG, reaching almost zero within the first few seconds for polynomials of order 9 and above. For EKG with its higher number of common peaks between two different people, the fall in violations is more gradual and does not fall significantly before 14th order polynomials and about 600 seconds time difference between two PSKA iterations. Finally, as expected for both EKG and PPG the higher the order of polynomial, the more the number of common features needed and therefore lower the chance of getting $v + 1$ common features between physiological signal measurements at different time stamps.

We can thus see that PSKA meets all our design goals. The PPG-based PSKA requires a smaller polynomial order and shows more time variance compared to EKG-based PSKA. Interestingly, the temporal variance plots for EKG and PPG also illustrate the level of synchronization required between sensors when either is used as the physiological signal of choice. We see that for both EKG and PPG the violations are highest when the time difference between the iterations is around 1 second, irrespective of the polynomial order used. This means that the features measured one second apart have not changed considerably thereby still allowing successful unlocking of the vault. We can therefore say that even if the communicating sensors measure their physiological signals for PSKA a second apart (even longer for EKG) they will succeed is agreeing on a common key.

[4]We considered over 100 random start-times for PPG and EKG, over all the ten subjects to compute the average violation

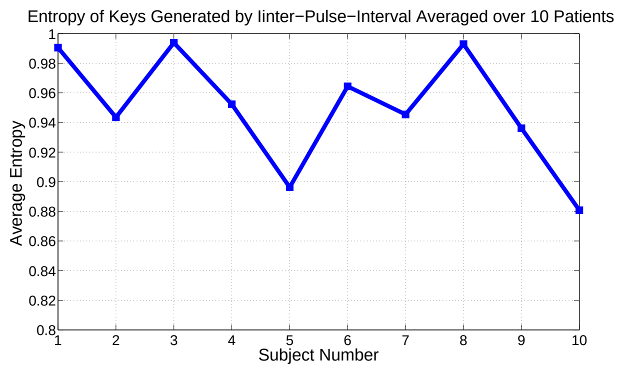


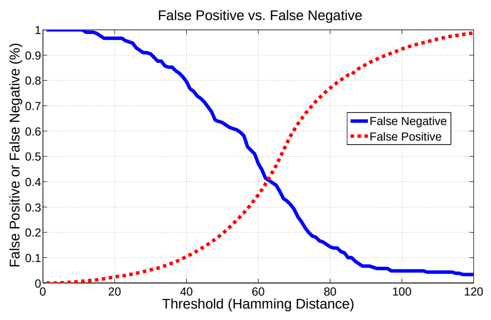
Fig. 15. Average Entropy of Keys generated from IPI

3.8. Related Work

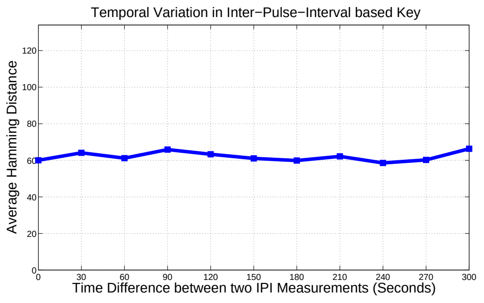
Key Distribution Protocols: Many key distribution protocols for sensor networks have been proposed in the literature. These protocols can be grouped into three categories based on the type of communication they secure: pairwise key distribution for securing unicast communication, such as [20] [91] [35] [39] [63] [67] [68] [89] [142], group-wise key distribution for securing multicast, such as [16] [18] and [142], and network-wise key distribution for secure broadcast such as [89] [114] [142]. In order to secure this communication, each time the network is setup each sensors have to be programmed with a secret key with their neighbors. This process would be executed by the doctor during the deployment of the sensors on the patient, or by the patient themselves. This tends to be a tedious and manual task.

Environmentally-coupled Key Generation: The idea of using physiological signals for securing inter-sensor communication was first introduced by us in [23] [130]. The principal idea was to use physiological signals for hiding the actual key to be shared between the sensors, and correct the differences in the physiological signals using simple error correction scheme. The paper however does not suggest any particular physiological signal which can be used. Building upon this initial idea, the authors in [93] proposed the use of Inter-Pulse-Interval (IPI) to generate cryptographic keys. The advantage of using IPI is that it can be derived from multiple sources namely Photoplethysmogram (PPG) and Electrocardiogram (EKG) time series by measuring the time difference between the peaks in the EKG/PPG signal. One can envision two sensors in a BAN designed to measure EKG and PPG respectively being able to communicate securely with one another using IPI based keys. The IPI-based key generation process works as follows: 1) the sensors first measure EKG and PPG signals in a synchronized manner; 2) they then generate a series of IPI values from their respective data; and 3) they take 67 contiguous IPI values from a particular start-point and encode them into 128 bit to form the key_{ekg} and key_{ppg} at each of the sensor which can then be used for secure communication between them. We conducted our own experiments of IPI based key generation using EKG and PPG data of ten subjects collected from the PhysioBank database (http://www.physionet.org/physiobank/). We found that even though the keys are long and random (see Figure 16(a)), the average hamming distance between key_{ekg} and key_{ppg} for the same subject is ∼60 bits, and ∼65 bits for two different subjects. The scheme proposed in [93] is therefore at best suited for *authentication purposes only* (as identical keys would be needed if the key is to be used for encryption and message authentication code). This can be easily done by performing the following check $Hamm(key_{ekg}, key_{ppg}) < Threshold$, where $Hamm()$ is the function for computing the Hamming distance, and $Threshold$ is the Hamming distance threshold between the two keys. Figure 16(b) shows the false positive[5] and false negative[6] rates of IPI based authentication, and we can see the results are not encouraging. We believe the primary reason for this difference is the *topographic specificity* of the human body — physiological signals measured from different areas of the body appear to have similar trends (high correlation) but not the exact same values. As a result, the information symbols in the keys get re-ordered leading to translational and rotational errors [54] which produce drastically different values. Additionally, IPI requires quantizing 67 IPI values for generating a key, which means about 30 seconds of EKG/PPG measurements are required (as an EKG/PPG peak is generate every 300-500 msec) before keys can be generated, which is not adequate for real-time requirements of the BAN. Also, the keys generated from IPI lack temporal variance. This is important, as the knowledge of the key_{ekg} or key_{ppg} at any given time should not allow adversaries to authenticate successfully at a later time. It can be seen from Figure 16(c), the average Hamming distance between the keys generated for a person over time remains around 60 bits. This means that if the current value of a key_{ekg} or key_{ppg} is known and the threshold used is 60 bits (based on

[5]when key_{ekg} and key_{ppg} of two different people are less than $Threshold$
[6]key_{ekg} and key_{ppg} for the same person are above $Threshold$



(a)



(b)

Fig. 16. a) False Positive and False Negative while using IPI for Authentication, b) Time Variance of IPI-based Authentication Keys

the Figure 16(b)) or above, the probability of using it to authenticate during any subsequent authentication attempts is close to one [125]. In [75] [76], the authors present a Candidate Key Protocol for generating pairwise cryptographic keys from feature vectors generated from accelerometer data. Their schemes require the involvement of the users, in terms of physically shaking the two devices, and pressing a button (called “authenticate now”) on the devices to synchronize the accelerometer signal measurement process and execute the protocol. We believe that CKP though an environmentally couple security solution cannot be used for BANs, because:

- CKP depends upon the accurate shaking of the two devices in order to be able to establish a secure channel between them. If not done carefully, the whole process has to be repeated. With PSKA, such a problem does not exist as the physiological values used by it cannot be consciously controlled by humans and therefore, will always appear identical (baring topographic specificity which can be corrected) to the two the sensors measuring them.

- Sensors in a BAN are expected to work autonomously, involving users in enabling secure communication between them as CKP does (the need for shaking the device) considerably reduces the usability of the security solution for BAN environments.

- Given the size and the continuous data collecting nature of the BANs, expecting users to be involved in establishing a secure communication channel between each pair of sensors which want to communicate also makes its very inefficient for BANs.

We therefore claim that CKP is not suited for BANs even though it works with environmental characteristics as PSKA because of its workflow. This is not surprising as CKP was designed for hand-held devices such as cell phones not sensors in a BAN.

Fuzzy Vault: In [54], the authors suggest the use of a fuzzy vault which has the ability to tolerate the re-ordering of information symbols. The scheme is ideal for approaches dealing with physiological signal features, as two samples of any physiological signal are never completely identical due to the dynamic nature of the body. The fuzzy vault scheme has so far been primarily applied to biometric based authentication using finger-prints [122] and iris-images [97]. However, the non-variant biometric template opens up the fuzzy vault to attacks involving template modification [107]. Similarly, in [58], the authors present the implementation of an attack on fuzzy vault system used for biometric authentication systems. The attacker intercepts two vaults generated from the same biometric (fingerprint) data with different chaff points, correlates them to reveal the hidden biometric features. The both the attack succeed because the fingerprint features in vaults do not change. These attacks are not possible with PSKA because the feature values in the fuzzy vaults generated in two iteration of PSKA are drastically different due to the temporal variance property of the physiological signals used. Therefore trying to tamper with a vault or correlating two vaults will not give the attacker any information in PSKA.

In [22] the authors propose a method to reduce the effort required by an attacker to find the legitimate points from a fuzzy vault. They propose the notion of free area of a point in the vault which is defined as the positions in the vault where the new point can be placed. The authors argue that for two points $p1$ and $p2$ in the vault, if free area of $p1$ is greater than the free area of $p2$ then the probability of $p1$ being added to the vault earlier than $p2$ is higher than the points being added the other way around. An attacker, by ordering the points in the vault in decreasing order of free area can distinguish between the legitimate and chaff points in a vault (the legitimate points in the vault would be the ones which were added to the vault the earliest and therefore have higher free area compared to chaff points added later). However, by their own admission they could not prove this claim and therefore try to give empirical evidence to its validity. We believe that the attack proposed in this work is more of a mathematical exercise rather than a genuine attack on the vault from the perspective of the attacker. The authors have simply computed the probability of a point having high free area being a genuine point in the vault by trying out thousands of different vaults and state there is a “good” probability associated with this fact. In order to compute this probability they fill the vault keeping track of which points are genuine and which are not, compute the free area of the points, and see in how many instances the genuine points had high free area. Their results show that for vaults with very large set of features points (165, 330 and 495 points) the probability of successfully stating that a point with high free area is genuine approaches 0.5 (Figure 4 in the paper), which is essentially random. With PSKA the number of features is in the range of 30 for PPG and 80 for EKG, the probability of a point having high free area being a legitimate point will be minuscule. The attack proposed in [22], therefore, does not provide an attacker with any distinct advantage in identifying legitimate points from the chaff over a brute-force attack with respect to PSKA.

In [79] the author suggests a brute-force attack on the fuzzy vault which bounds the number of operations required to guess the legitimate points in fuzzy vault, with a high probability, using a polynomial of order v to $8vlog^2v(|R|/|N|)^v$. The consequence of this result is that with a probability close to 1, the complexity of identifying the polynomial used by the vault decreases. Using our parameters R = 5000, N = 30, v = 9 for PPG based PSKA is now only as secure as brute-forcing a 75 bit key rather the original 95 bit key (based on our previous estimate, where we were trying to compute the upper bound of brute-forcing the vault; the complexity of identifying $v+1$ points on the chosen polynomial with probability equal to 1), still fairly strong. If we increase the polynomial order to 12, the strength goes up to 97 bits. Our EKG results which uses R = 5000, N = 87, v = 12 provides a security of about 80 bits. However, given the larger number of feature points (t) the overall security provided by EKG is lower than PPG, keeping everything else constant. Further, in both cases, if we increase the number of chaff points (our feature length is 13bits which allows for a range of 0 - 2^{13}) we can again increase the complexity of breaking the vault. Of course, increasing the polynomial order or vault size is going to increase the memory usage and communication complexity, but given that the length of the key exchanged using PSKA is arbitrarily long, we do not have to execute PSKA regularly and thus should be affordable.

4. SECURING INFORMATION ACCESS IN SMART-INFRASTRUCTURES

4.1. Introduction

Recent years have seen the development of smart infrastructures which consist of a large number of heterogeneous, massively distributed computing entities. Such infrastructures provide their users with an aware, intelligent, information rich environment for conducting their day-to-day activities [6]. Examples of smart-infrastructures include - health monitoring systems [113] [129], smart-spaces [104], aware-homes [46] etc. An important application of their monitoring capabilities is *emergency management*. Examples of emergencies include patient needing urgent medical attention, crisis such as building fire, and the computing infrastructure under attack from outside. Smart-infrastructures can be used to detect, provide useful and real-time information about the state of such emergencies to the planners and relief-workers and facilitate response, thereby improving the chances of saving lives and property.

Traditional wisdom dictates that in the event of emergencies, any security system in place be disabled in order to allow relief workers to utilize full capabilities of the system for controlling the emergency [78]. Such an approach may work for traditional non-smart systems and infrastructures but given the extent of sensitive information available within smart-infrastructures, disabling security in the event of emergencies, will potentially leave the system vulnerable to hackers and tech-criminals. For example, consider the scenario where a person faces a health related emergency, and the security on his wearable health monitoring system is disabled so that any doctor or medical personnel arriving at the scene can easily view the subject's health information without any security constraints. However this also allows any malicious entity in the vicinity, with the right set of tools, to access the person's health information as well. Similarly, it is also possible that malicious elements may dupe the smart-infrastructure into detecting a false emergency, disable the security system and collect sensitive information from the space. We therefore contend that while utilizing smart-infrastructures for emergency management, care has to be taken that the privacy of the infrastructure and its users is protected. We define *privacy preservation during emergency management* as temporarily providing the necessary information and services for responding to the emergency, to specific subjects, until either the emergency is contained or it cannot be controlled any more. But before we discuss the details of how to enable privacy preserved emergency management, we present some of the principal concepts regarding emergencies and their management. In the rest of this chapter we use the term smart-infrastructure and system interchangeably.

4.2. Preliminaries

The data dissemination for emergencies requires a dynamic access control model. Many access control solutions exist for securing access to medical data for clients such as [98] [25]. These models provide access to the patient's health information in a reactive manner, that is, access is provided only on explicit request from the caregivers. Though adaptive, they only take into account the current context of the users, before providing appropriate access. What they do not consider is the occurrence of critical events in the system and providing access for this change [132]. In order to better understand this behavior expected from access control models which work in emergencies, we present some preliminaries concerning the concept of access control for data dissemination, criticalities and their properties, the system model - which describe some of our principal assumptions with respect to its operation, and design goals for CAAC.

4.2.1. Emergencies or Criticalities: Concepts

Emergencies also known as *criticalities* can be defined as a consequence of specific events called *critical events* on a system. Critical events are those whose occurrence moves a system into an abnormal/unstable state. Criticalities usually require timely *response actions* to be *controlled* - their adverse effects negated. A criticality which has not yet been controlled is called an *active criticality*. In practical terms, controlling a criticality means minimizing the possibility of loss of lives, services and property in the event of a criticality. Each criticality has a time duration associated with it known as the *window-of-opportunity* (Wo), within which response actions have to be taken for the criticality to be controlled. A criticality is controlled only if all the response actions for it are executed within the Wo. A criticalities whose Wo has passed is said to be *expired*. *Criticality management* is defined as a process by which the effects of the criticality can be controlled. For example, consider the criticality due to a person having a heart-attack. In this case, the criticality management are the steps that need to be taken to preserve the person's life, help in recovery, and ensure management of the same criticality in the future. The knowledge of this process will help us understand how access control can be utilized to aid in the process. As illustrated in Figure 17, criticality management has four phases:

- *Detection* - this phase is responsible for detecting criticalities in a timely manner in order to ensure they can be controlled within their Wo. For example, if the criticality is a heart-attack, it can be detected based on chest pains and shortness of breath.
- *Response* - this phase specifies the actions that need to be taken to bring criticalities under control, and protect lives and property. The principal aim here is to control the immediate symptoms seen during detection. For a heart attack, response actions may include calling 911, providing access to patient's past health data to paramedics, taking the patient to the hospital, administering CPR, and so on.
- *Mitigation* - once the immediate effects of criticality are contained, this phase deals with long term recovery efforts. During mitigation, the heart attack patient is admitted to the hospital and monitored by a personal doctor for relapse and complications.
- *Preparedness* - this phase deals with analyzing the criticalities of the past and get ready for the future ones. It is executed during system planning phase or when the system has experienced criticalities which have now been controlled. For example, during preparedness phase, one can evaluate the effectiveness of the detection (patient feedback on chest pain) and the need for improved methods (automated monitoring), or possible alternate response actions (and their order) which might reduce the response time to well within the Wo. The preparedness phase is also executed before the system is deployed in order to determine the effectiveness of the steps that need to be taken at the other three phases.

4.2.2. Single and Multiple Criticalities

In [44] we introduced the notion of access control for handling criticalities. The principal idea of the chapter was to temporarily promote the roles of specific subjects within the system in the event of a criticality. However the work assumed that at any given time, a system suffered from a single criticality, which needs to be controlled within its Wo. In reality a system seldom suffers from single criticality, the occurrence of one emergency, most of the times leads to others, all of which have to be responded to with their respective Wo. For example, a building catching fire is a criticality and the presence of trapped people within this burning

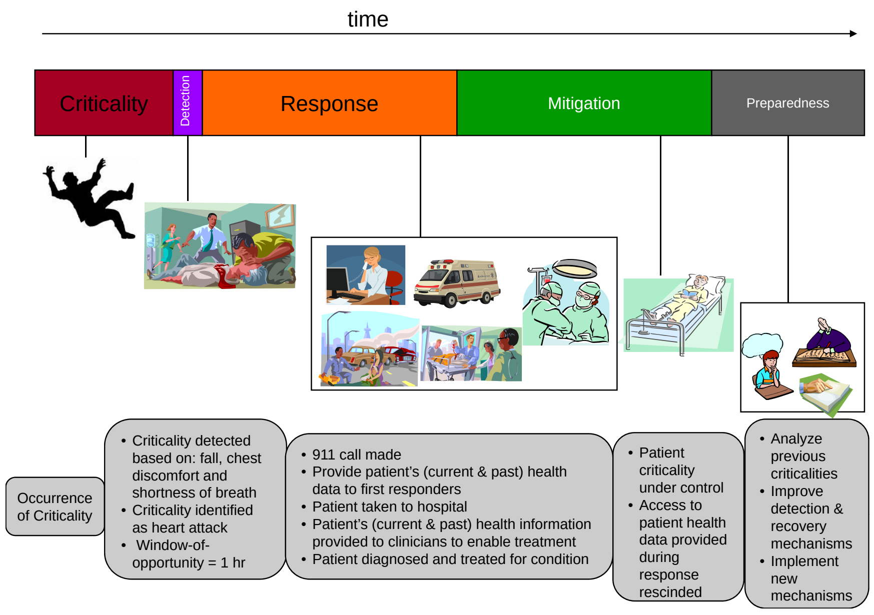

time
Criticality
Detection
Response
Mitigation
Preparedness
Occurrence of Criticality
• Criticality detected based on: fall, chest discomfort and shortness of breath
• Criticality identified as heart attack
• Window-of-opportunity = 1 hr
• 911 call made
• Provide patient's (current & past) health data to first responders
• Patient taken to hospital
• Patient's (current & past) health information provided to clinicians to enable treatment
• Patient diagnosed and treated for condition
• Patient criticality under control
• Access to patient health data provided during response rescinded
• Analyze previous criticalities
• Improve detection & recovery mechanisms
• Implement new mechanisms

Fig. 17. Criticality Management Phases

building is an additional criticality, both of which need to be controlled (i.e. building saved, and people rescued).

Handling multiple criticalities is considerably more complex than single criticality for many reasons.

- We have to keep track not only of the occurrence of new criticalities but also control and expiration of existing criticalities.

- This requirement is further compounded by the fact that occurrence of criticalities, and the response actions required to control them have a stochastic nature due to the probability of human error in executing them [83]. Therefore it is possible that responding to a criticality might lead to other criticalities within the system. The determination of response actions at any time have to take these into account.

- With multiple criticalities, we have to prioritize one criticality control over the others, such that the probability of control of all the criticalities is maximized. Therefore, depending upon the combination of criticalities present in the system, the response actions required to be control them may vary. For example, if a person is facing an high blood pressure, the principal response action is to give them the required medication. If however the person now develops angina pains, it may be required that the angina be controlled first before the person is administered medication for hypertension, in order to increase the chance of her/his survival.

Systems designed to handle multiple criticalities have to be aware of all these requirements while they facilitate response actions.

4.3. Our Approach

In this work we present a novel access control model called **Criticality Aware Access Control** (CAAC) [44]. CAAC is a dynamic model which has the ability to respond to the occurrences of emergencies. CAAC takes an *environmentally-coupled approach to access control* to achieve its goal by facilitating a more **adaptive and proactive** access control in the event of emergencies within the system. The principal goal of CAAC is to be aware of its surrounding and provide the right set of privileges for the right set of subjects, at the right time for the right duration to facilitate emergency response.

The criticality-awareness provided by CAAC is fundamentally different from context-awareness that many traditional authorization models possess. Context-awareness takes into account the contextual information of the subject making the access request in deciding whether access should be granted or not. With criticality-awareness on the other hand, the contextual information considered is for the whole system and all its components (not the subject making the request alone). This system context is evaluated continuously and in case of a deviation from the norm, appropriate privileges are provided for subjects to deal with it, even without an explicit request from the subject. Criticality is therefore a context that has a direct consequence on the system itself (not just on the subject making the request) and therefore needs to be dealt with significantly differently (in a proactive manner) compared to traditional context awareness.

But, before we describe CAAC and its constructs we present some preliminaries regarding emergencies, their properties and what is required by criticality aware access control systems.

4.4. System Model

The CAAC model views the world as consisting of two two groups objects and subjects. *Objects* are entities both physical and virtual which provide a variety of information and services. In the case of a smart-infrastructure they are all the entities which provide services and information, example, printers, laptops, even objects such as doors and elevators. *Subjects* on the other hand are entities which access the information and services provided by the objects. The CAAC model controls attempts by subjects to access the objects within the environment. The access control model is deployed and managed by an *administrator* whose is trusted. Further, the system is assumed to have an efficient logging capabilities which are assumed to be non-tamperable and like a flight-recorder keep detailed logs on all the activities within the system, especially during criticalities. Techniques such as [110] [73] can be used for this purpose.

All access control systems need an underlying authentication system to function. The model assumes that the smart-infrastructure provides it with authentication facilities which allows to reliably identify subjects. Any RFID based challenge/response to biometric based authentication [137] [14], smart card based authentication [11] can be used here. Note that the technology used by the authentication system is not the focus of this work; we simply assume that it has the ability to authenticate subjects reliably and provide this information to the access control system in place. Finally, we assume that all criticalities are detected reliably, that is, their types and properties are accurately known at the time of detection, using techniques such as [66] [69].

4.4.1. Design Goals

In terms of security properties, this is mainly an *authorization problem*. Access control models are typically used to authorize access to specific information and services for subjects in the system during day-to-day activities. We contend that for smart infrastructures, they can be easily applied for privacy preserving emergency management, by using them to *facilitate response actions* to bring criticalities under control. Access control models therefore can be used in the response phase of criticality management process. In this regard, keeping in mind the requirements of handling multiple criticalities in a system, we have identified two basic *design goals* that our access control approach needs to posses:

- **Adaptiveness :** Access control approaches for criticality management have to be able to - 1) determine the response actions required for handling the current set of criticalities within the system, and 2) change the privileges available to the subjects in order to maximize the chances of controlling all the criticalities within their Wo.

- **Proactivity:** Access control approaches for criticality management should be able to determine the subjects for executing the response actions, and enable them to execute the required set of response actions (even those which are not allowed during normal operations); thus avoiding the need for any explicit access requests, in an accountable manner.

We have identified five metrics, which characterize the two design goals. They are:

- *Correctness* - It ensures that the response actions are facilitated by CAAC only as a consequence to the occurrence of critical events within the system.

- *Liveness* - It requires that any access privilege provided in response to critical events is only for a finite amount of time.

- *Responsiveness* - It ensures that occurrence of criticality facilitates response actions, which requires provision of access privileges to the right set of subjects, and subject notification.

- *Non-repudiability* - It mandates that all response actions taken within the system during criticality are recorded for accountability purposes.

- *Safety* - It ensures that only authorized change to resources and access control constructs can happen within the system.

We contend that the first three metrics are a necessary conditions for demonstrating CAAC's adaptiveness, while the last three demonstrate its proactivity. Intuitively, if CAAC is adaptive, it has to determine the privileges needed that to be provided for managing a criticality (responsiveness), temporarily change the access privileges to subjects based on state of criticalities change within the system (correctness, liveness).

Similarly, if CAAC is proactive, it has to be able to legitimately authorize subjects with any privileges in order to enable them to take take response actions (safety), without any explicit access request (responsiveness), in an accountable manner (non-repudiation).

This is fundamentally different from how access control mechanisms have worked thus far. Traditional access control techniques such as [106] [105], [139], [88] [49] which provide access to information in a reactive manner, that is, access is provided only on explicit request from the subjects in the system. With criticalities, response actions need to be fast in order to meet the Wo requirements. Therefore, waiting for request from subjects can arbitrarily slow down the response actions. Further, they do not have any notion of the stochastic nature of criticalities and the response actions needed to control them. Additionally, they use simplistic event based rules for changing the privileges to subjects, which limits their adaptiveness for criticality management.

4.5. Criticality Aware Access Control (CAAC)

CAAC is a complete access control solution for smart-infrastructures. It specifies the privileges that subjects get on various objects in the system under both normal situations and criticalities. In normal situations, the default privileges are provided, while during criticalities it facilitates temporary alternate privileges during the response phase which may provide subjects with different capabilities (greater or lesser than before) than they normally would have to enable criticality response [124].

The CAAC model uses the basic constructs of Role Based Access Control (RBAC) model [106] for controlling access for subjects (S) to objects (O) in the system. We choose RBAC as the basis of CAAC as we want to demonstrate how criticality awareness can be incorporated in an existing, widely used access control model. In CAAC, subjects in the system have a role associated with them. Roles (R) are representation of subjects' responsibilities, and is assigned to them when they become part of the system. Subjects can have many roles, but can activate only one at a time. Examples include a doctor joining a hospital being assigned the roles of a *surgeon* and *doctor of patient X*. Even though a subject can have many roles, they can activate only one at a time. The CAAC model keeps track of the current role the subject has taken and provides provisions for switching between the roles. To determine the actions that subjects can perform on specific

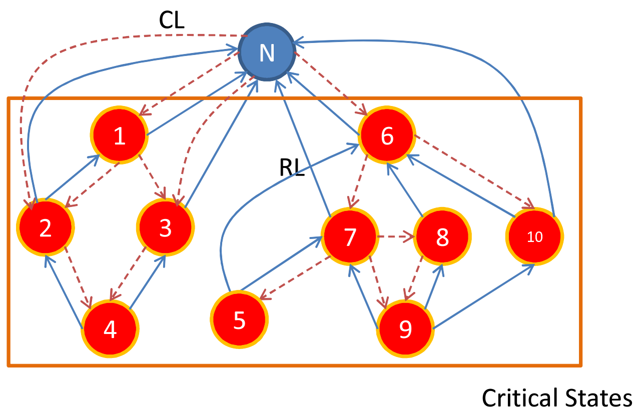


Fig. 18. Example Action Generation Model

objects their roles are indexed into an Access Control List (ACLs) maintained by objects in the system. ACLs are tables defined for each object in the system which maps roles to associated privileges. Privileges (PR) are authorizations which allow subjects to execute specific actions within the system. For example, ability to read specific sensor data, actuating a treatment and so on.

CAAC's execution is based on a set of policies (P). Initially, when CAAC is being setup the administrator of the system establishes the set roles, privileges and their default mappings (ACL) that will be available in the system, along with the policies. Under normal situations, when a subject wants to access an object in the system, they make a request to the access control model. The request includes subject's role and context information along with the specific privileges required. The model indexes the subject's role to the object's ACL and checks to see if the requested privileges are equal to the associated privileges in the ACL. Note that access provision need not be completely static. System specific contextual information associated with the subject can be used to vary the privileges. Under normal situations CAAC behaves similar to a context aware access control model proposed in [49] [27]. When the system experiences criticalities (C), the access control model becomes more proactive in nature. It evaluates the nature of the criticalities in the system, identifies the response actions that need to be taken and proactively enables them.

4.5.1. Criticality Response Facilitation in CAAC

In this section we present how CAAC facilitates the response actions. It has three phases - identifying the response actions, identifying subjects to take the actions, and temporarily providing the chosen subjects with the privileges to execute the response actions, and rescinding the privileges when they are not needed.

4.5.1.1. Identifying Response Action

The *preparedness phase of the criticality management process* is used to determine the set of actions for all possible combinations of criticalities that can occur within the system. This is not such a leap because criticalities need to be known and their effects understood before they can be responded to. Such steps are taken in the design of all systems these days. For example emergency procedure manuals are developed for managing common emergencies such as fire and earthquakes in buildings. The current techniques for determining the actions to be taken during criticalities are qualitative in the nature, which though sufficient for CAAC are not ideal as we cannot consider all possible combination of criticalities in the system and evaluate their effects. Further, having quantitative models can allow them to be evaluated in a computer and their results available electronically in a format understood by CAAC. Therefore, once we have identified the criticalities possible within the system, we can enable the best response actions for them using CAAC.

Action Generation Model : We have developed a novel and effective way of determining the response actions for criticalities within the system called the *Action Generation Model* (AGM), based on the criticality modeling framework defined in [81]. It consists of two types of states - normal and critical states. When a criticality occurs, the system moves from the normal state to a critical one. The system's state keeps changing as new criticalities occur or get controlled. The system reaches one of the normal states only when all active criticalities in the system are controlled. The transitions which take the system toward the normal state are called *Response Link* (RL) and those that increase the number of active criticalities in the system are called *Critical Link* (CL). Each CL and RL has a probability associated with it. Figure 18 shows an example AGM. The dashed lines are the CLs while the full lines are the RLs. The sum of the probabilities of all the CLs and RLs originating from a given state is 1. The CL associates with the probability of a particular criticality occurring given the current state of the system. In the figure, CL between States $< 1 >$ and $< 2 >$ specifies the probability of the occurrence of a criticality which takes the system to State $< 2 >$ when the system was originally in State $< 1 >$. Similarly, taking a RL represents the execution of the associated one or more response action(s), taking the system from a lower state toward the normal state, and the probability of successfully executing them (based on human error probabilities). In the figure, RL between States $< 2 >$ and $< 1 >$ specifies the probability of the successfully executing the response actions which takes the system to State $< 1 >$ when the system was originally in State $< 2 >$. Each RL and CL also has a time associated with it, which determines the time taken to take that link, for CL it signifies the time for criticality to occur and for RL it is the time to take the response actions. From a given critical state, the system may have multiple links (RLs) which can potentially take it toward the normal state. For example from State $< 2 >$, the normal state can be reached by at least 2 links - one through the link to State $< 1 >$, and the other a direct link to the normal state. Each RL represents a different set of response actions, which if taken in the current state, can potentially lead the system to the normal state.

Response Action Metric and Planning using AGM : The principal use of the AGM mode is to determine the best RL from each critical state so that the probability of reaching the normal state is maximized. From a given critical state, the choice of a particular RL depends upon its *Q-value*, which is a combination of three factors: 1) The probability of successfully reaching the neighboring state from the current state, by taking the RL; 2) The probabilities of successfully reaching the normal state from the neighboring state; this is compiled as an aggregate value by considering all possible paths to the normal state from it; and 3) ensuring that window-of-opportunity of all the active criticalities in the system are met at all times. For a RL to be chosen, it has to have the maximum Q-value, as this represents the best response actions, given the current state the system is in. For example, if the system is in State $< 4 >$, then chosen RL is given by $max(\{p(4,2) * P(2,N)\}, \{p(4,3) * P(3,N)\})$ provided the Wo for all the criticalities active in State $< 4 >$ never expires. Here $p(i,j)$ is the probability of reaching State $< j >$ from State $< i >$, and $P(k,N)$ is the aggregate probability of reaching normal state from State $< i >$ through all possible paths to N, for $P(2,N)$ we have $p(2,1) * p(1,N) + p(2,1) * p(1,3) * p(3,N)$.

Identifying the RL to be taken from each system state, also by definition identifies the response actions that need to be taken if the system ever comes to that state. The aforementioned computation of Q-value of each RL and choosing the one with the maximum value, provides the optimal solution. We call this the *optimal planning criterion* [83]. However, using Q-value as the basis of identifying the next RL suffers from two problems -1) the state space explosion in computing the Q-value, and 2) the computation of Q-value returns zero, if the Wo of any of the active criticalities expire, in which case, no RL is returned from the current state. To overcome this problem, we use two *heuristic planning criteria* - 1) choosing the RL with Maximum Probability at the current state, and 2) by choosing the RL at the current state whose actions take the Minimum Time. Using the heuristic criteria, not all criticalities in the system may be controlled, but some of them may be. The execution of AGM off-line means that such situations can be detected early and can be avoided by designing better response actions, faster criticality detection mechanisms in order to maximize the probability of reaching the normal state from a critical state. If for some reason, it is not possible, we can use a mixture of optimal and heuristic planning criteria for different states of the AGM. The principal goal is enable actions such that, even if the normal state is unattainable, the number of criticalities in the system is reduced.

Criticality Response Evaluation Tool : In order to automate the AGM execution process, a *Criticality Response Evaluation Tool* (CRET) is used. The tool was first described in [82], and takes as input the set of states, the CLs, the RLs, and the optimal or greedy rules of determining the best RL, and depending upon the current state the system, determines the best RL (and consequently response actions) for that state. CRET uses a XML based specifications for specifying the inputs. However, it lacks one capability, in that it does not allow the specification of response actions associated with each RL. It only has the ability to specify the time taken to execute the response actions associated with a RL. We have therefore extended the XML specifications of CRET to allow us to specify the response actions associated with each RL.

Using CRET during the preparedness stage, planners can provide the CAAC model with the appropriate response actions which need to be taken given the current state of the system. During the operation of the system, CAAC checks it periodically (every tp time units) to determine the current state of the system. If it is critical, CAAC determines the RL (and therefore response actions) to take based on the optimal planning solution using the AGM which was executed during the preparedness stage (using CRET). If the optimal planning criterion returns null (Q-value becomes zero) for any state, it simply chooses one of the heuristic (greedy) planning criteria to determine the RL that needs to be taken from that state. The idea is to control as many criticalities in the system as possible. If the system was previously in a critical state and has now moved to another one, then a new set of actions are required to move the system to the normal state. Given the actions that need to be taken, the CAAC can enable their executions by providing appropriate subjects with the privileges (specified in the actions). Determining the subjects' ideal for handling the emergency is the next phase of CAAC. An important question at this point is how the response actions should be specified.

Response Action for CAAC : Each RL enables the control of a specific criticality within the system. Taking a RL means executing the response action associated with it, which in turn controls the criticality. As there may be multiple ways of responding to a criticality, the question now arises as to which one to choose as the response action associated with a RL. There can be many (system dependent) factors which determine the response action. The process of choosing the response action to be associated with a RL has to be first and fore-most *risk-averse*, taking into account the following factors: 1) probability of success, 2) knowledge of the number and capability of subjects who may execute the actions, and 3) the availability of resources to pursue the action. The relative priority associated with these factors is again system dependent. For example, if the criticality is a ventricular fibrillation on a subject with implanted pace-maker the possible task associated with responding to it could be - 1) command the pace-maker to shock, and 2) use external defibrillator. If we assume that both actions require one subject and the necessary tools are available, and (1) has 90% success rate compared to (2) whose success rate is 10%, we associate the action 'command the pace-maker' with RL whose probability is now 0.90.

In order to facilitate the response action thus chosen, it is specified as a tuple with two elements $a_i =< ObjectID, Privileges >$, the object on which the action is to be executed and the privileges required to on them. For example, in our previous example, object in question is the *defibrillator* and the privilege is *use*. As the object-privilege tuple enables only a specific action, it can be thought of as following the *principal of least-privilege* in facilitating the response action. We choose this model of action representation as it mimics CAAC's understanding of response actions which we will describe in the Section 4.5.2. We illustrate a specification of the response actions, in the case study in Section 4.7. In general, each RL is associated with a task set $TS = \{a_1, a_2, ...a_k\}$, where k is the number of response sub-actions of the form described earlier. The privilege associated with a RL will be provided to the chosen subjects (see next section) even if they conflict with the subjects' default privileges in the system, as criticality response is of paramount importance for the functioning of the system.

4.5.1.2. Subject Selection

The primary difference between CAAC and traditional access control models is that it identifies the action that need to be executed within the system (in response to a criticality) and solicits subjects to perform those actions by providing them with appropriate privileges. Identifying subjects who can perform the response actions is therefore an important requirement for CAAC to function. We posit that subject selection in CAAC is primarily a function of criticality. This is understandable because the nature of the problem usually decides the number and type of people required to address it. For example, fire fighters are required to handle fire criticality, cardiologist is needed for cardiac criticalities and so on. Given the knowledge of the types of criticalities within the system during the preparedness phase, subject selection for taking response actions can be primarily done in two ways:

- *Statically :* Depending upon the criticality, the set of subjects required for controlling it can be predetermined and stored in a static list. For example if the possible criticality is a fire, then the subjects required for controlling it are fire-fighters in the nearest fire station. Such an approach is best suited for criticalities which require subjects who are not part of the system but are well known. Note that, as the identity of the subject is known before hand, the response action for a RL (out the many possibilities) can be chosen such that the privileges which enable it follow the least-privilege principal (i.e. deviate from the default privileges of the subject by a minimum).

- *Dynamically :* Maintaining a static list for all criticalities does not work in situations where it is not possible to know the identity of subjects beforehand. For example in the event of a medical emergency in an ICU any doctor on duty should be given the privileges to view the patient's data irrespective of whether she is the patient's assigned doctor or not. To handle such situations, rules could be defined which specify contextual criteria (location = ICU AND Status = "On-Call") that need to be met by the subjects in order to be selected.

Once the subjects are chosen, they will be provided the privileges required to execute all the response actions for that state. This is done in order to give the chosen subjects the flexibility to coordinate among themselves and perform the required actions in an efficient manner. The CAAC model thus de-links the choice of subject selection from the action generation, which allows it to easily handle criticalities in the system. With traditional access control models, implementing such a scenario would require enumerating all possible privileges available to subjects under different sets of criticalities [124].

4.5.1.3. Enabling Response Actions

Once the current action that needs to be taken at the current system state has been identified, and the subjects to execute the actions have been chosen, the actions need to be enabled by providing appropriate privileges, the subjects notified, and the privileges rescinded at a later time. This is done as a three step process:

- *Alternate Privileges :* The first steps in enabling response action is to provide the chosen subjects with alternate privileges. This can be done by assigning them with a new temporary role and adding a new entry to the appropriate object's ACL. The new entry will provide the privileges required for taking the actions required at this time. As the underlying role of selected subjects has been modified, they cannot at this time perform any of their traditional tasks unless that is part of the response actions. CAAC keeps track of the mode it is functioning in and in the presence of criticalities does not allow any non criticality response access request.

- *Inform Subjects :* Once the appropriate changes to subjects' role and objects' ACL have been made, CAAC informs the subjects of their new capabilities as well as the information about the criticalities at hand and the actions that need to be taken. At this point the system maintains detailed records of the alternate privileges provided to subjects and the actions that were taken with these privileges.

- *Rescind Privileges :* The system is designed to periodically check its state to determine the current set of response actions as we saw previously. During each of these checks, depending upon the state the system is in; a new set of response actions is generated (based on CRET execution in preparedness stage). If the set of actions generated are different from the ones that are being enforced at this point, we have to rescind the current set of privileges and provide new ones to enable the new set of response actions. The privileges thus provided in the previous states are rescinded once they are no longer required. We do this to avoid conflicts between any privileges provides in the past and those provided for responding to current criticalities within the system. Such rescinding of privileges also take place if the system has reached the normal state (no criticality) or that criticality is beyond control, in which case the system has to be audited based on its logs.

Enabling subjects to take response actions is akin to *delegation of authorization* where the delegator is not a subject in the system, but the system itself. Once the actions have been executed or no longer needed the delegation is rescinded, which is exactly what happens with CAAC. However, CAAC does not allow for delegation chains as some of the other access control models do with the objective of maintaining strict control over criticality response. Delegation chains have the ability to considerably improve the flexibility in taking response actions, with the side-effect of increasing the complexity of the system considerably. This is because, allowing people not chosen by the system to be provided with alternate privileges can not only result in misuse but also increase the chances of error in response actions (RL probability) than what is used in AGM, leading to incorrect RL choices.

4.5.2. Policy Specifications

Given the constructs of CAAC, we are now in a position to describe the policies which are used to implement it. The policies are described in the guarded command language form where a sequence of guards are followed by sequence of actions [32] [105], represented as:

$$\textit{guard} \longrightarrow \textit{command}$$

A policy rule is read as: if *guard* is true then execute *command*. The guard is usually a predicates which must hold before the command is executed. The guard may represent the contextual parameters such as occurrence of events or presence of specific conditions - time, user characteristics (location, designation). The command on the other hand usually specifies the action (specified directly, through predicates or system dependent functions) that need to be taken if the contextual cues in the guard are satisfied. In some cases, the commands can be themselves be predicates. We use the notation $X.a$ to refer the attribute a of a basic component X. The predicates used to describe the guards are of the form $p_1 \wedge p_2 \wedge ... \wedge p_n$, where p_i is a predicate of the form $predname(c_1, c_2, ...c_n)$, where c_i is a basic component's attribute. Basic components form the building blocks of the policy specification. Each basic component is of the form $<x_1, x_2, ...x_n>$, where x_i characterizes the component, and can be a constant, variable, another basic component.

4.5.3. Principal Policy Specification Components

The following are principal basic components used for the CAAC policy specification.

1. *Role (R)* - Each roles is a variable which abstract out the responsibilities of subjects in the system. It is usually user-defined and represented as $Role =< role >$.

2. *Privileges (PR)* - Define the authorizations that can be associated with entities in the system. Privileges are represented as authorizations ($PR =< auth >$), where authorization $< auth >$ can take one of the following as its values read, write, use. The authorizations are system dependent, and more can be added if required.

3. *ACL* - Is an access control list, defined as a collection of tuples of the form $ACL =$ set of $< r, p >$, where $r \in R$, and $p \in PR$, which define the authorizations associated with specific roles. An ACL is always associated with an object.

4. *Subject (S)* - These represent the subjects in the system are represented using the tuple S = set of $< sId, X >$, where $\forall g \in X, g \in R$ and sId is unique subject id of type $< string >$.

5. *Object (O)* - These are the entities in the system on which privileges are provided for the subjects, and are defined using the tuple O = set of $< oId, ACL >$, where oId is unique object id of type $< string >$, and ACL is the associated access control list.

6. *Criticality (C)* - These are the criticalities that can occur within the system are defined using the tuple C = set of $< cId, Wo >$, where cId is the unique identity of the criticality, Wo is the variable storing its window-of-opportunity.

Apart from the basic components, the CAAC model also maintains a set of tables for managing its various functions. These include:

1. *Subject-Role Table (SPR)* - SPR maintains the currently active role of the subject. Each entry of SPR is a tuple of the form $< s, r >$ where $(s, .) \in S$ and $r \in R \wedge r \in S.X$. Further, $(s, .) \in S$ denotes that subject with sId s is one of the subjects in the system. The function $currentRole(c)$ returns the role associated with subject c in the SPR table.

2. *Static-Subject (SS)* - SS maintains the list a static list of subjects who need to be selected in the event of a particular criticality. SS are represented using a tuple of the form $< c, x >$ where $c \in C$ is a criticality that can occur within the system and $x \subset S$ is the identity of the set subjects who are required for responding to the criticality.

3. *Dynamic-Subject (DS)* - DS maintains the list a set of rules which identify the subjects who need to be selected in the event of a particular criticality, based on their current context. Each entry in DS is a tuple of the form $< c, t >$ where $c \in C$ is a criticality that can occur within the system, t is a predicate which specifies the contextual criteria for the subjects to meet to be chosen. Any context or group of contexts can be specified as a part of the t using the mechanism given in [49].

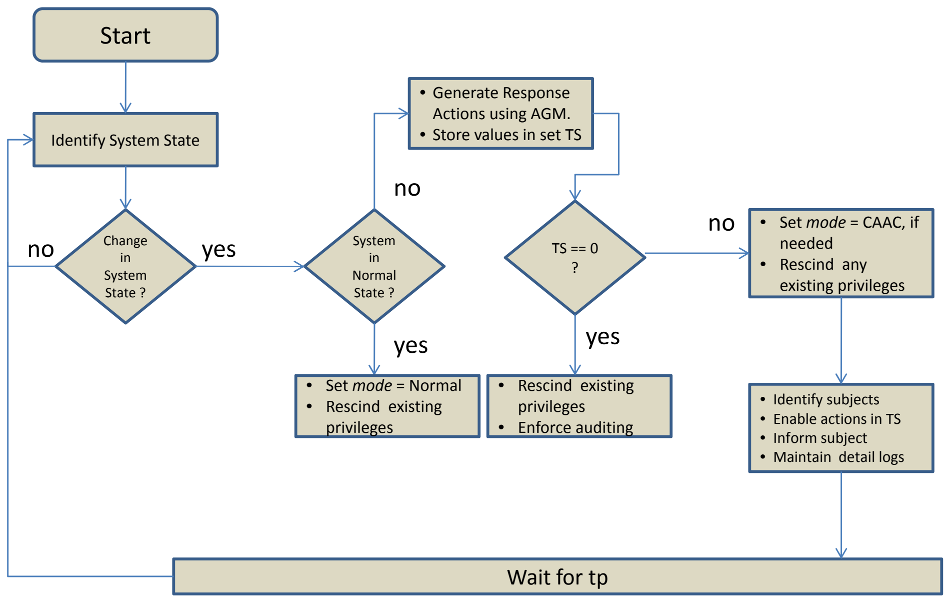


Fig. 19. CAAC Criticality Management Process

4. *Old Role (OL)* - OL is the table used to store the original roles of the subjects who have been given alternate privileges in response to a criticality. Each entry of OL is a tuple of the form $< s, r >$ where $(s, .) \in S$ and $r \in R$. Using OL, CAAC can revert the roles of subjects to their original pre-criticality value, when the alternate privileges need to be rescinded.

The policies in CAAC can be categorized in to three main parts - administrative task control, access control and criticality control. We now define each of these and their associated policies in detail.

4.5.3.1. Administrative Control Policies

These are the policies which are used to perform the basic functions of the CAAC model such as adding and removing subjects, associating and dissociating subjects with roles, updating ACLs and so on. We list only the main CAAC policies here. Policies for removing subject, removing entries from ACL are trivial extensions and are not listed for brevity reasons. Each of the policies can be executed only by the administrator of the smart-infrastructure.

1. **Add Subject** - Adds a subject (s) to the system, along with the set of roles that the subject can take. The policy can be executed only by the system administrator. We assume the presence of a system-

dependent function $Auth()$ is used for this purpose which checks to see if the subject claiming an id really has access to it.

$AddSubject(s, roles, u_{admin}) \wedge Auth(u_{admin}, Admin) \wedge \forall r \in roles,\ r \in Role \longrightarrow S = S \cup \{s, roles\}$

2. **Activate Role** - Activates the role of a subject by storing it in a specific subject-role (SPR) table.

   $ActRole(u, u_{admin}, r) \wedge (u, .) \in S \wedge Auth(u_{admin}, Admin) \wedge r \in u.X,\ r \in R \longrightarrow SPR := SPR - \{u.sId, .\}, SPR = SPR \cup \{u.sId, r\}$

3. **Add ACL** - adds a new ACL entry to an object.

   $AddACL(r, p, o, u_{admin}) \wedge Auth(u_{admin}, Admin) \wedge (o, .) \in O \wedge p \in PR \longrightarrow if \nexists \{r, p\} \in o.ACL$ then $o.ACL := o.ACL \cup \{r, p\}$

4.5.4. Access Control Policy

This access control policy (ACP) is used to evaluate the access request of specific subjects and provide the requested privileges if the request holds true.

$$ACP(u, o, priv) \wedge (u, .) \in S \wedge (o, .) \in O \wedge priv \in PR \wedge \{currentRole(u), priv\} \in o.ACL \wedge$$
$$(\text{if}(mode == true)\text{then}(uContext(u) == oContext(o))\text{ else true}) \longrightarrow true$$

Here the function uContext(u) returns the current context of the subject u, while Context(o) returns the context expected by the object (o). The context is evaluated only under normal situations and not during criticalities.

4.5.5. Criticality Control Policies

These policies are used for enabling the CAAC model to control the criticalities that exist within the system. There are three main policies which accomplish this task, which we describe below:

1. **Alternate Privileges** - Provides the alternate privileges required to enable criticality management. It utilizes the task set TS generated by the $getTS()$ function for the current state the system is in, as described in the Section 4.2. TS is defined as $TS = \{d_1, d_2, ...d_k\}$, where d_i is an action of the form $< o, pr >$, where $o \in O$ and $pr \in PR$. It also needs a list of chosen subjects Sub who will be granted

the alternate privileges. The original privileges of the subjects are stored in the table OL, which will be used to reset the subject's role.

$AltPriv(Sub, TS) \wedge \forall x \in Sub,\ (x, .) \in S \wedge \forall TS.d_i,\ (TS.d_i.o, .) \in O,\ TS.d_i.pr \in PR \longrightarrow \forall TS.d_i,\ AddACL(r_n, TS.d_i.pr, Ts.d_i.o, u_{admin}),\ \forall x \in Sub,\ OL := OL \cup (x, currentRole(x)),\ ActRole(x, u_{admin}, r_n)$

2. **Inform Subjects** - It is used to inform the chosen subjects (Sub) they have alternate privileges. We utilize a specific function $Inform$ for this purpose.

   $InformSub(Sub, TS) \wedge \forall x \in Sub,\ (x, .) \in S \wedge \forall TS.d_i,\ (TS.d_i.o, .) \in O,\ TS.d_i.pr \in PR \longrightarrow \forall TS.d_i,\ Inform(Sub, TS.a_i.pr\ , Ts.a_i.o, u_{admin})$

3. **Rescind Privileges** - This policy rescinds the alternate privileges provided earlier in the event of a change in the number of criticalities or elimination of all criticalities from the system.

   $RescPriv() \wedge \forall TS.d_i,\ (TS.d_i.s, .) \in S,\ e.r \in R \longrightarrow \forall e \in OL,\ ActRole(e.s, u_{admin}, e.r)$

4.5.6. Execution Model

In this section we present the execution model of CAAC. Figure 19 shows the flowchart which illustrates the CAAC model's criticality management process. Figure 20 gives the pseudocode for the entire process. The model runs in an infinite loop monitoring the system for change in system state. The system dependent $checkSystemState()$ function is used for this purpose. If a change is detected, we check if the system is now moved away from its normal state. If so the system is moved to the CAAC mode. In the CAAC mode, given the current state of the system, the function $getTS()$ returns the TS by simply mapping the state to the result of the off-line CRET execution. Once TS is known, the set of subjects necessary to carry out the actions are selected based on the SS and DS tables. Each of the selected subject is provided with the privileges to execute the actions in TS using the $AltPriv$ policy, and then informed using the $InformSub$ policy. All actions taken when the system is in the critical state is recorded using the $RecordActions()$ function. The function is system dependent and we therefore refrain from defining it here. The recording is required to ensure accountability to the CAAC model to ensure that any illegal activity performed using the alternate

```
CAAC_EXEC()
stateChange := false
selSubject := $\phi$
Mode := Normal
while(true)
    t := checkSystem State()
    if(t $\neq$ currentState)
        currentState := t
        currentCrit = findCrit(t)
        RescPriv()
        if(t $\neq$ Normal)
            mode := CAAC
        else
            mode := Normal
        endif
    endif
    if (mode == CAAC)
        TS = getTS(currentState)
        foreach (e $\in$ SS)
            if (e.c == currentCrit)
                selSubject := selSubject $\cup$ {e.X}
            endif
        endforeach
        foreach (d $\in$ DS)
            if (d.c == currentCrit)
                foreach (s $\in$ S)
                    if (uContext(s) == d.t)
                        selSubject := selSubject $\cup$ {s
                    endif
                endforeach
            endif
        endforeach
        if (AltPriv(selSubjects,TS))
            InformSub(selSubjects,TS)
            RecordActions()
        endif
    endif
    Wait $t_p$
endwhile
```

Fig. 20. CAAC Execution Model

privileges for criticality control are detected. Once the privileges have been provided, the system waits for tp duration of time, and repeats the whole process, again. If the system moves from one critical state to another, then any alternate privileges provided earlier are rescinded using the $RescPriv()$ function.

4.6. Compliance with Design Goals

CAAC is supposed to be proactive and adaptive in nature and is designed to provide right access control to right set of subjects at the right time for the the right duration, in order to control un-controlled criticalities within the system. In Section 4.4.1, we had defined a set of metrics which demonstrate CAAC's adherence to its design goals of proactivity and adaptiveness. In this section we present semi-formal proofs to illustrate the CAAC policy specification's adherence to the design goals. We assume with respect to our proofs that the access policies are implemented and enforced correctly.

Theorem 4.6.1 Responsiveness : *When a critical event occurs - 1) the subject is immediately notified, and 2) its access privileges are changed.*

Proof The proofs of the claims above are as follows: 1) When there is a criticality, the subjects are notified in Line 33 of Figure 20. 2) The role of the subject being notified is updated in Line 32 of Figure 20 using the policy $AltPriv$. The policy preserves the current role of the subjects in the table OR for resetting when needed Line 9 of Figure 20 using the $RescPriv$ policy. ∎

Theorem 4.6.2 Correctness : *Subjects in the CAAC framework get alternate set of privileges iff there is at least one un-controlled criticality in the system.*

Proof If there is at least one un-controlled criticality in the system, the $mode$ variable is set to $CAAC$ in lines 10 - 15 in Figure 20. This results in the execution of Lines 16-34 in Figure 20, thus providing alternate set of privileges to subjects. If a subject is allowed to execute actions which are enabled by alternate privileges, it means the function the $mode$ is set to $CAAC$. As this can happen only if the $currentState$ of the system is in critical state showing the existence of at least one criticality in the system. Hence, the result follows. ∎

Theorem 4.6.3 Liveness : *The maximum duration for which subjects are assigned alternate privileges is limited by - the time instant when the number of active criticalities in the system changes.*

Proof From Theorem 4.6.2, it follows that subjects receive alternate privileges iff there is at least one criticality in the system. The alternate set of privileges for subjects are changed by invoking in Lines 9 in Figure 20 to rescind existing privileges and providing new ones by executing Lines 32 of Figure 20. Both these actions can be executed only if there is a change in the current state of the system (Lines 6, Figure 20), which can happen only if there is a change in the number of active criticalities (occurrence of new one or control of existing one) in the system. ∎

Theorem 4.6.4 Non-Repudiation : *Malicious use of alternate privileges when system is experiencing criticalities is non-repudiable and limited to a finite amount of time.*

Proof Line 34 in Figure 20 ensure that whenever a role is changed and a request for resource is successful, it is recorded along with the appropriate start and end times enforcing non-repudiation of any malicious activity by a subject due to the new privileges. As we assume that all the access control policies execute correctly, the records are accurately updated. From Theorem 4.6.2 and 4.6.3, it follows that subjects are granted alternate privileges only in the presence of criticalities and the maximum time for which the alternate privileges are granted is limited by the time when a change in the number of criticalities occurs, thereby limiting potential malicious activity to a finite amount of time. ∎

Theorem 4.6.5 Safety : *Only authorized access is allowed for 1) objects and 2) access control constructs within the model.*

Proof 1) Access to resources is allowed, only if ACP evaluates to true. Now as in both normal and critical situations subjects access to an objects is only by CAAC by invoking the ACP, authorized access is ensured. 2) Access for modifying constructs of the access control model can be done by executing any of the CAAC administrative policies. Each of these predicates can be executed only by CAAC using the administrator role (U_{admin}), thus ensuring authorized access to the access control model.

We can see that CAAC satisfies each of the five metrics set forth earlier in the chapter. We therefore claim that CAAC is both proactive and adaptive in nature thus meeting its design goals.

4.7. Case Study: Emergency Access Control on Smart Oil Rig Platforms

CAAC is a versatile access control mechanism for criticality management. In this section we demonstrate how CAAC can be used for criticality response in a larger more complex systems. We consider a scenario of a medical and fire emergency in a smart-infrastructure - a smart oil rig. We chose oil rig as an example because of the availability of the probability values for the AGM to determine the response actions.

We look at a combination of two types of criticalities in this example - 1) health-related and 2) fire accident. The goal is to show the ability of CAAC to handle any conceivable combination of criticalities in a system. We assume that the oil rig is a smart-infrastructure where each individual subject can interact seamlessly with their environment to obtain services. The smart-environment also keeps track of the subjects within the rig. The entire rig is managed by the Rig Management Environment (RME) which provides the common interface for subjects to interact with. We divide our example into two parts - preparation and

TABLE 4

Criticalities and Properties

Criticality ID	**Criticality**	Wo	**Task-Set (TS)**	**TS Exec. Time**
c1	Heart Attack	5 min	Defib: $< -, -, Yes >$ Health Data:$< True, True, - >$	1 min
c2	Fire	20 min	Fire-Ext: $< -, -, Yes >$	2 min
c3	Unstable Angina	60 min	Health Data:$< True, True, - >$	1 min
c4	Fire Assistance	12 min	CR Door: $< -, -, Yes >$	2 min

execution. The former, deals with executing the AGM using CRET to determine response actions, while the latter deals with the access control during emergencies.

4.7.1. Preparation

Before the rig is deployed, in order to handle criticalities within this environment, planners and engineers of the oil-rig will have to execute the criticality response evaluation tool (CRET) in order to determine the actions which need to be taken during specific emergencies. Four possible criticalities within the oil rig are considered for this example -

- **c1** - A worker on the rig with a chronic hypertension having a heart attack in the control room.
- **c2** - Fire alarm in the control room of the rig.
- **c3** - A worker on the rig with a chronic hypertension having unstable angina in the control room.
- **c4** - People trapped in control room needing immediate assistance.

The Table 4 provides details for each criticality and its important characteristics - window of opportunity and the response action task set required to respond to each of the criticalities and the time taken to execute the response actions. The extended CRET XML schema is used to specify our scenario. Specification 1 shows a sample state transition specification, here $Object - ID = 3$ is the fire extinguisher. Figure 21 (a) shows the AGM that is generated by CRET using the stochastic model described in Section 4.5.1.1. The probabilities of the CLs and RLs have been obtained from various studies on occurrences of emergencies and response errors for medical ailments [21] [94] [59] [47] and oil rig management [33]. The CRET is then executed to determine the response actions for each critical state in the system using the optimal planning criterion for choosing RL at each state (based on Q-value). If the Q-value for a state returns zero, then any of the greedy planning criteria (MP or MT) is chosen to control as many criticalities as possible.

Figure 21 (b) shows comparative results, in terms of Q-value, on probability of successfully controlling all the active criticalities within the system. It can be seen that as the number of criticalities in the system increases, the lower the probability of reaching the normal state. Further, as the number of states in this example are limited, the optimal solution and the heuristics yield similar Q-values. Also, for lower states (those with large number of criticalities), the RL chosen by the greedy approach yields zero Q-value (i.e zero probability of controlling all criticalities within the system) as the window-of-opportunity is not satisfied for at least one of the active criticalities in the system if the RL chosen by the greedy approach is taken in that state.

The execution of CRET gives us the next link from each critical state in order to have the maximum probability of reaching the normal state. These collections of links can be viewed as a path to the normal state. Figure 21 (c) shows the paths yielded by the optimal and greedy criteria. Note that, even if the Q-value may be identical for the optimal and greedy approaches, the actual path and therefore the response actions that need to be taken may be different as can be seen from Figure 21 (b) and (c). For some critical states, the optimal planning criterion returns a zero Q-value. The heuristic planning criteria provides some set of actions in all cases, but there is no guarantee that all criticalities in the system will be controlled, as in the case of states 8 and 9 for both MT (chooses ML with minimum response action time) and MP (chooses ML with maximum probability in the state). Given the path, the planners can easily identify the set of actions that need to executed, and the order in which they have to be executed, to ensure that the criticalities present within the system can be controlled. For example, using the optimal criteria, the path from State $< 5 >$ goes to State $< 2 >$ and then to the normal state. Therefore, in order to have a high probability of reaching the normal state, one needs to first take actions to respond to the fire and then respond to the heart-attack. Apart from determining the task set for a specific criticality, the planners also determine other requirements for specific criticalities such as subjects that need to take response actions. As mentioned previously, subjects can be selected based on their capabilities (statically) or based on their contexts (In this example we consider only

Specification 1 CRET State Transition Specification for Fire Criticality

```
              ⋮
<TransitionMatrix>
       <BeginState>3</BeginState>
       <EndState>1</EndState>
       <Probability>0.34</Probability>
       <Time>2</Time>
       <TaskSet>
              <NumberOfTasks>1</NumberOfTasks>
              <Actions>
                     <ObjectID>3</ObjectID>
                     <Privileges>
                            <Read>-</Read>
                            <Write>-</Write>
                            <Use>Yes/Use>
                     </Privileges>
              </Actions>
       </TaskSet>
</TransitionMatrix>
              ⋮
```

the location context for simplicity reasons.). The response actions and subjects to be chosen are then used to populate the CDL specifications used by CAAC. Specification 2 shows CDL representation of the response actions for CAAC when it is at State $< 2 >$. Here $ObjectID = 1$ is the health-data, and $ObjectID = 2$ is the defibrillator as shown in Specification 3. The specifications for the other policy components have similar form and are therefore not included for clarity reasons.

4.7.2. Execution

Given the response actions that need to be taken at each critical state in the system, let us now illustrate how CAAC functions. Figure 22 shows the access control constructs used by CAAC for performing access control in this scenario. The tables on the left side of the rig show elements in normal state, while the ones on the right show the values during criticalities. Consider the scenario were a hypertensive crew member (Geologist,ID-X) in the control room has had a heart attack ($c1$). The CAAC model which is routinely evaluating the system state (every t_p time units) notices that the system is not in the normal state any more but in the critical state (State 2 in the AGM). It determines that the path to reach the normal state is by responding to the criticality $c1$ directly. The task-set for this has two actions - enable defibrillation, and provide access

Specification 2 CDL Response Actions Specification

```
<Response-Actions>
       <StateID>2</StateID>
       <Task-Set>
              <NumberOfTasks>2</NumberOfTasks>
              <Actions>
                     <ObjectID>1</ObjectID>
                     <Privileges>
                            <Read>True</Read>
                            <Write>True</Write>
                            <Use>-/Use>
                     </Privileges>
              </Actions>
              <Actions>
                     <ObjectID>2</ObjectID>
                     <Privileges>
                            <Read>-</Read>
                            <Write>-</Write>
                            <Use>Yes/Use>
                     </Privileges>
              </Actions>
       </TaskSet>
       ⋮
</Response-Actions>
```

to X's health information. CAAC then checks the SS and DS tables to identify the best subjects to take the actions. This is the *rig-medic*. The CAAC then changes the role of the rig-medic to a temporary *CAAC-role*, updates the ACL of objects - defibrillator, and health data of X with the new role which has the appropriate privileges (see Figure 22). It then informs the rig medic about the changes, who can take the required action.

If before the arrival of the medic, a fire breaks out in the control room ($c2$), then the system is now in State $< 5 >$ of the AGM. At this state, the CRET execution demands that first the fire be responded to and then the heart attack be addressed. CAAC further determines that all the subjects within the range of the control room - i.e. those in infirmary, cabins and control room be given the privileges to control the fire (Technician,ID-3; Rig-Manager,ID-4; and Rig-Medic,ID-1 in our example). Further, the task set for criticality $c2$ states that the object fire extinguisher be provided access to. CAAC provides the privileges to all the people in the three locations with access to the fire-extinguisher object, by changing their active role to CAAC-role (the old roles are stored in the OL table) and adding an entry to fire extinguisher object's ACL (see Figure 22). It then informs all the chosen subjects about the changes, who can take the required action.

Specification 3 CDL Object Specification

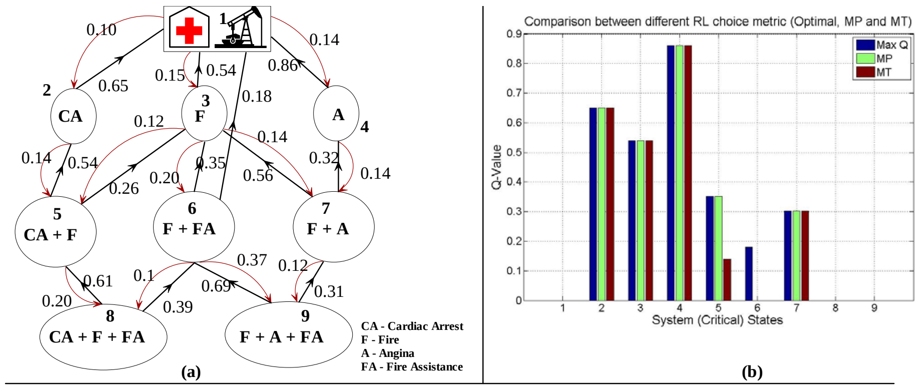
```
       ⋮
<Object>
       <ObjectID>2</ObjectID>
       <Description>Defibrillator</Description>
       <AccessControlList>
              <Role>All</Role>
              <Privileges>
                     <Read>-</Read>
                     <Write>-</Write>
                     <Use>No/Use>
              </Privileges>
       </AccessControlList>
</Object>
       ⋮
```

As the state of the system changed, before providing privileges for fire fighting, it rescinds the privileges of the rig-medic for handling $c1$, in order to enable them to fight the fire which is far more dangerous to many more people. Once the fire is controlled the system moves back to State $< 2 >$ in the AGM, and the rig-medic is provided the privileges for accessing X's data and using the defibrillator as he sees fit. The privileges for fire-fighting are rescinded, and the old roles of subjects involved are returned (except rig-medic who is now the chosen subject for $c1$). Once the heart-attack is controlled as well, the system is in the normal state and the role of the rig-medic is also changed to the default value. If suppose the X's heart-attack cannot be controlled within its Wo, then the privileges of rig-medic are rescinded by changing their roles to the original values. All the actions taken during the criticality response are recorded for determining the effectiveness of CAAC. The example thus demonstrates how CAAC can be used for managing criticalities within the system by providing the right set of privileges, to the right set of subjects, at the right time for the right duration.

4.8. Related Work

In this section we look at some of the prominent classes of access control and discuss how their utility for privacy preserved criticality management.

Access Control Models: Over the years, many access control models have been proposed for enabling authorized release of information. One of the most influential is the *Role Based Access Control (RBAC)*. RBAC was first thoroughly studied in the seminal chapter by Sandhu et al. [106]. This chapter defined the basic



Optimal
State 2: 2 -> 1
State 3: 3 -> 1
State 4: 4 -> 1
State 5: 5 -> 2 -> 1
State 6: 6 -> 1
State 7: 7 -> 3 -> 1
State 8: Not Controllable
State 9: Not Controllable

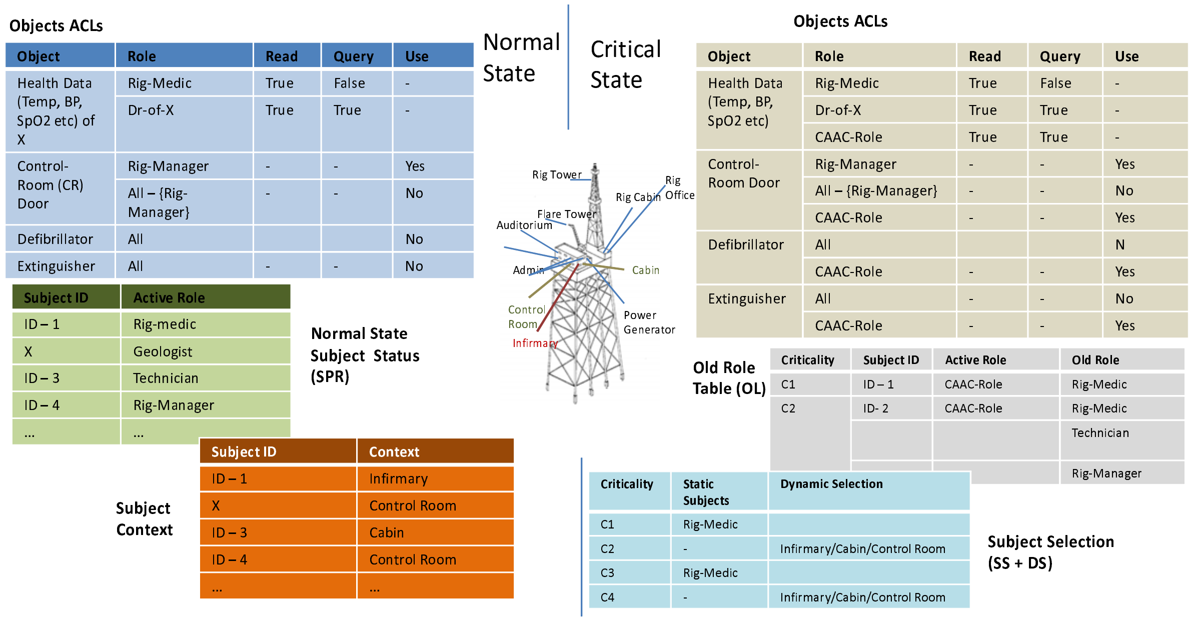
Greedy (MP)
State 2: 2 -> 1
State 3: 3 -> 1
State 4: 4 -> 1
State 5: 5 -> 2 -> 1
State 6: 6->3 -> Not Controllable
State 7: 7 -> 3 -> 1
State 8: 8->5-> Not Controllable
State 9: 9->6-> Not Controllable

Greedy (MT)
State 2: 2 -> 1
State 3: 3 -> 1
State 4: 4 -> 1
State 5: 5 -> 3 -> 1
State 6: 6->3 -> Not Controllable
State 7: 7 -> 3 -> 1
State 8: 8->6-> Not Controllable
State 9: 9->6-> Not Controllable

(c)

Fig. 21. CAAC Example - a) AGM for Medical and Fire Criticalities in Oil Rig, b) Criticality Response Q-Values at each critical state, c) Path to Normal State for each Critical State

components of RBAC such as subjects, roles, and privileges, their interactions (constraints and hierarchy). By de-coupling the process of directly associating privileges with a subject, RBAC provided an effective and easy way of managing security and enforcement of complex access control policies within the system. The concept of RBAC was generalized in [80] by incorporating subject roles, object roles and environment roles. As most systems have dynamic requirements, RBAC was further extended by including different types of context information in the access control decision making process, leading to the developing of *Context Aware-RBAC (CA-RBAC)*. In [27], present a CA-RBAC which considered the spatial, temporal and resource context in access control decision making, [42] presents team based access control model which was context-aware, [52] analyzes the temporal considerations for the basic RBAC model. Newer paradigms of access control which divert from the Role-based approach, such as *Context Based Access Control (CBAC)* [26], and *Usage Control (UCON)* [134] [88]. CBAC works by associating privileges directly with context information for each subject and avoids the notion of roles increasing the simplicity, while UCON combines the notions



Objects ACLs (Normal State)

Object	Role	Read	Query	Use
Health Data (Temp, BP, SpO2 etc) of X	Rig-Medic	True	False	-
	Dr-of-X	True	True	-
Control-Room (CR) Door	Rig-Manager	-	-	Yes
	All – {Rig-Manager}	-	-	No
Defibrillator	All			No
Extinguisher	All	-	-	No

Subject ID	Active Role
ID – 1	Rig-medic
X	Geologist
ID – 3	Technician
ID – 4	Rig-Manager
…	…

Subject ID	Context
ID – 1	Infirmary
X	Control Room
ID – 3	Cabin
ID – 4	Control Room
…	…

Objects ACLs (Critical State)

Object	Role	Read	Query	Use
Health Data (Temp, BP, SpO2 etc)	Rig-Medic	True	False	-
	Dr-of-X	True	True	-
	CAAC-Role	True	True	-
Control-Room Door	Rig-Manager	-	-	Yes
	All – {Rig-Manager}	-	-	No
	CAAC-Role	-	-	Yes
Defibrillator	All			N
	CAAC-Role	-	-	Yes
Extinguisher	All	-	-	No
	CAAC-Role	-	-	Yes

Criticality	Subject ID	Active Role	Old Role
C1	ID – 1	CAAC-Role	Rig-Medic
C2	ID- 2	CAAC-Role	Rig-Medic
			Technician
			Rig-Manager

Criticality	Static Subjects	Dynamic Selection
C1	Rig-Medic	
C2	-	Infirmary/Cabin/Control Room
C3	Rig-Medic	
C4	-	Infirmary/Cabin/Control Room

Fig. 22. CAAC Example - Access Control Structures

of access control, trust management and digital rights management to provide finer grained access control to a subjects who may not be known to the system. None of these schemes were designed for privacy preserved criticality management. They are reactive in nature and wait for subjects to ask for specific privileges. Basic RBAC and its derivatives are static in nature and do not have ability to change privileges of subjects. Context-aware version of RBAC, UCON and CBAC have the ability to change the privileges available to subjects, but they are done based on context changes for the subject making an access request. This is too simplistic for criticality management considering its stochastic nature, the need for urgent response actions and human error involved in executing them.

Optimistic Access Control (OS): Traditional access control schemes have been static in nature and evaluate access requests based on fixed set of rules. In [96], the authors present the notion of *Optimistic Security*, which takes the approach that most access within the system are legitimate and relies on detailed auditing procedures for ensuring that the security policy is maintained. The scheme lets subjects exceed their default privileges but in a semi-constrained manner in that 1) it records all actions taken during the time when subjects exceed their privileges and 2) it allows subjects to execute only those actions which can be rolled back. Similar schemes have been presented in [100] and [41] as well. OS based schemes, are similar in approach to those provided the CAAC model, in that, it allows subjects (through alternate privileges) to perform actions which require privileges exceeding their default values. However, it does not control the duration for which alternate privileges are provided. Further, it requires human intervention (from the superior of the subject who needs the alternate privileges) for authorizing any alternate privileges that the subjects' request, making them unsuitable for criticality control.

Policy Based Smart-Infrastructure Management (PBM): Much work has also been done towards developing policy based smart-space management schemes. Such schemes allow for influencing the behavior of smart-spaces without hard coding the behavior into them. Policy based management schemes use some type of obligation policies to guide the actions of the system and take a event-condition-action view of things [72], [13]. In case of an event, and a pre-define set of system condition take a particular action. Work has been done on developing more expressive policy languages such as [30] [55], [70] and [115]. The policy based management schemes is adaptive but cannot be easily used to provide alternate privileges to subjects when needed. Further, their lack of awareness of the criticalities, the associated stochastic characteristics along with their event based triggering of change is too simplistic to suite the needs of criticality response, especially in the event of multiple criticalities. Further, they too have traditionally been reactive in nature, waiting for access requests from subjects before they allow or disallow any actions within the system, which may waste valuable time.

Policy Spaces (PS): Another access control model which possess some of the adaptiveness of CAAC, divides the policies into groups called *policy spaces* [8] which provide mandatory access, mandatory access denial and planned exceptions. The idea is that for specific situations, policies provide access which not normally allowed, just as in CAAC. However there is an important difference between the two. CAAC provides the alternative privileges proactively while policy spaces approach waits for subjects to request access and is thus reactive. In the event of a criticality, such an approach may waste valuable time.

Criticality-Oriented Access Control (COAC): The notion of altering access control privileges to enable criticality management for smart-spaces was first introduced in our preliminary work in [44]. The principle idea

TABLE 5

Comparison of Different Access Control Classes w.r.t. CAAC

Properties	RBAC	CA-RBAC	CBAC	UCON	OS	PBM	PS	COAC	CAAC
Proactive	No	No	No	No	No	No	No	Yes	Yes
Adaptive	No	No	No	No	Yes	No	Yes	Yes	Yes
Exceptional Actions	No	No	No	No	Yes	No	Yes	Yes	Yes
Single Criticality	No	No	No	No	No	No	No	Yes	Yes
Multiple Criticality	No	No	No	No	No	No	No	No	Yes

was to promote the role of specific subjects in the space, for a limited duration, for taking response actions. Further in order to prevent any malicious acts with the promoted role, detailed logs were maintained within the system for accountability. However, the scheme was limited in scope as it only addressed systems with single criticality. A stochastic modeling framework for evaluating the management of multiple criticalities in smart-infrastructures was described in [83]. It identified various conditions and properties for criticality management in smart-spaces, and provide a generic model for determining the response actions for a given set of criticalities within the space. However, it did not facilitate any access control. With CAAC, we combine the two ideas to facilitate response to multiple criticalities within the system. This is fundamentally different from our previous model, in that, it keeps re-evaluating the access privileges granted for the mitigation purposes in the system, as the number of criticalities in it change.

The Table 5 presents a summary of the capabilities of CAAC and traditional access control and CAAC (here the CA-RBAC access control scheme, includes temporal, spatial, or any other types of context-awareness).

5. PROTOTYPE IMPLEMENTATION

We implemented the PSKA and CAAC schemes on a Pervasive Health Monitoring System called **Ayushman** being developed at the IMPACT Lab. The principal aim was to demonstrate the viability of the two approaches on real systems. We first provide an overview of the test-bed architecture, and then describe in detail the implementation of PSKA and CAAC on it. We then conclude the section with a discussion on the lessons learned during the implementation process followed by an example operation scenario.

5.1. PHMS Conceptual Model

Before we delve into the details of Ayushman, we present the conceptual model on which it is based. The model provides an overview of the basic functionalities of a generic PHMS. Figure 23 presents the model. It consists of two planes: the **data gathering plane**, and the **data access plane**. The *data gathering plane* is used to collect medical data from patients and perform preliminary processing on it. It is made up of the Body Area Network of wireless health and ambient monitoring sensors, actuators and the base station. The sensors which constitute a BAN are heterogeneous in terms of capabilities and come in many forms. Some of the recent developments in this regard have produced sensors which are wrist wearable [71], implantable [62] [143] [111], as a part of ambulatory devices and biomedical smart clothes [86]. The base station on collecting patient data, performs local data processing before forwarding the data to the data access plane.

The *data access plane* consists of an infrastructure for managing the health data collected by the gathering plane. It provides facilities for organizing pre-processed health data into a structured electronic format (also known as *Electronic Patient Records (EPR)*), and storing them for the long-term. The data access plane is also used for reasoning about the data in the EPRs. Some of the features it can provide include detection of - the occurrence of a medical emergency, the failure of a specific treatment procedure, and inconsistencies between the proposed diagnosis and the symptoms. The data access plane is populated with multiple information systems implemented using computational devices such as PDA, cell-phones, PC, and servers depending upon the requirement. It connects the PHMS to the out-side world and provides an interface for various clients (caregivers / family) to access data collected by the PHMS.

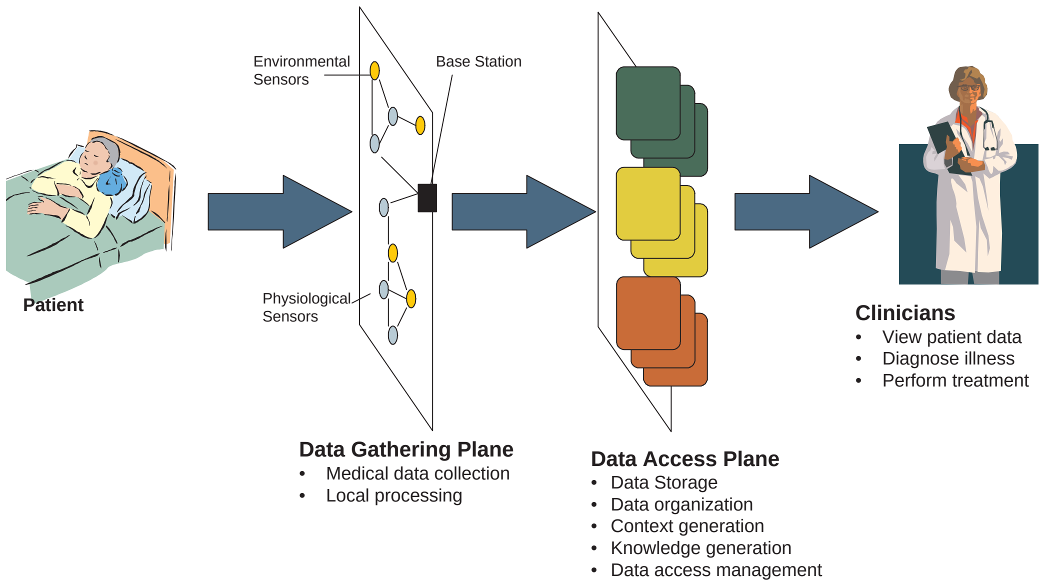


Fig. 23. A Generic Pervasive Healthcare Model

5.2. Ayushman Pervasive Health Monitoring System

The Ayushman system integrates heterogeneous components such as wireless sensors, medical devices, hand-held and larger computational devices to form a medical monitoring systems in an attempt the implement the PHMS conceptual model. Figure 24 shows its principal components, which we now describe.

Body Area Network is the most fundamental component of Ayushman and is responsible for collecting health data from patients. It is made of two types of entities: the sensors and the base station. Sensors collect data from the patients and are of two types:

- **Physiological Sensors** are wireless sensors which are worn by patients in order to collect physiological data from the patients.
- **Environmental Sensors** are wireless wearable sensors which measure the environment around the patient and measure parameters such as temperature, humidity and light. The environmental sensors provide the contextual information in the light of which the physiological data of the patient is evaluated.

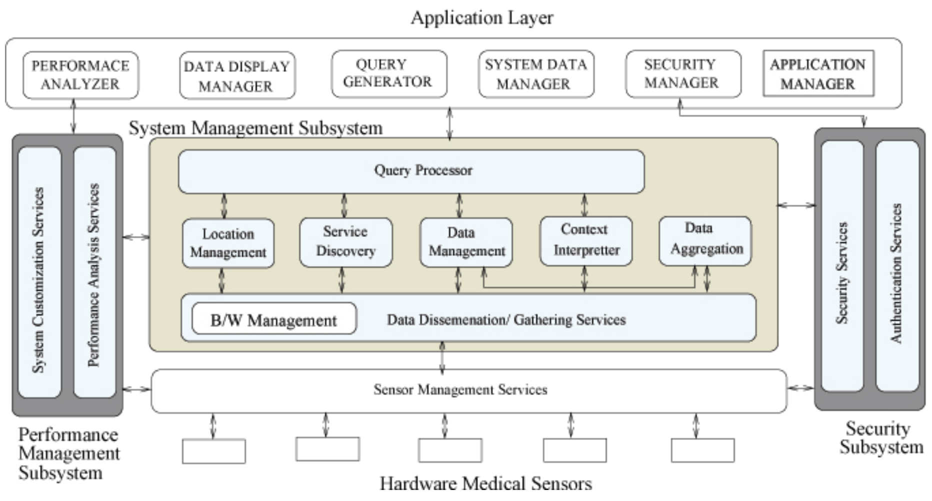


Fig. 24. Ayushman Architecture

The other entity in a Body Area Network is the *base station*. The base station is a highly capable computational entity which is carried by the patient. All the sensors on a patient send their data to the sink, which maintains a database for storing the data. A sink is typically implemented as a part of a hand-held device such as cellphone or a PDA. The use of a hand-held devices allows patients to be mobile. Further the spread of hand-held devices in recent years makes it quite more likely that patients possess one and keep in near them at all times. The sink always possesses the most up-to-date information on the patient's health. Sensors within the BAN form a multi-hop network to reach the sink. Depending upon the number of sensors in the network, a variety of topologies are possible including cluster, tree, linear etc. It provides a singular interface to the higher tiers of Ayushman through the sink. Data collected within a BAN around the person is forwarded regularly to the higher tiers for long term storage and analysis.

Back-end Systems are computational entities which populate the higher tiers of the Ayushman architecture, and collectively provide facilities for storage, analysis remote data retrieval. In our case we had a single tier system, where the data from the base station was directly sent to an *central sever* which contains data from the entire system. Depending upon the scale of the system, more number of tiers can be added to the network.

Remote Clients or Subjects are entities which connect the Ayushman system to query and receive past and current data from the patients. Remote clients connect to the central server through the Internet and query data from the specific patients. The central server returns the data if its local database has it or it forwards the query downward till a device is found which contains the data.

5.3. Ayushman: Implementation

In this section we describe our efforts in implementing the architecture starting with hardware used followed by the software implementation details.

5.3.1. Hardware

Ayushman is implemented using a diverse group of hardware technologies at both the front and back-end. The data collection is done using low-powered radios called motes developed by Crossbow Inc. We use mote models; MicaZ and TelosB. Both motes have a 8MHZ processor, utilize a ZigBee radio to communicate and are powered with 2AA batteries. TelosB motes have in-built sensors and are primarily used for environmental data collection. The data collected include: ambient temperature, humidity, and light. The MicaZ motes on the other hand are utilized for physiological data collection. These motes do not have in-built sensors therefore, they utilize additional hardware to provide us with physiological data. We use MicaZ motes to collect three types of physiological data: blood pressure, % of SPO2 and acceleration. Figure 25 shows different components in our setup.

We use pluggable (which utilize the 51 pin connector on board) sensor boards containing accelerometer for collecting acceleration data. The blood pressure and SPO2 sensors are external devices which are connected to the motes through the serial interface. The blood pressure device is built using the SunTech Advantage OEM board [4] while the SPO2 device is manufactured by SmithMedical [3]. To connect the BP device to the MicaZ mote, we built a interface board (IB2) (see Figure 25) which provided a DB9 interface for the BP monitor and which could be stacked on to the MicaZ mote using the 51pin connector. IB1 utilized a level shifter to switch between the RS232 logic levels and the MicaZ logic levels. We implemented an additional interface board (IB1) (see Figure 25) to connect the SPO2 device to the MicaZ mote. The design for this board was obtained from Harvard University CodeBlue project [113]. We could not use the IB2 here because the SPO2 device is not self powered like the BP device and needed to be powered by the mote.

The data from these motes is collected by the base station. We have implemented the base station on a

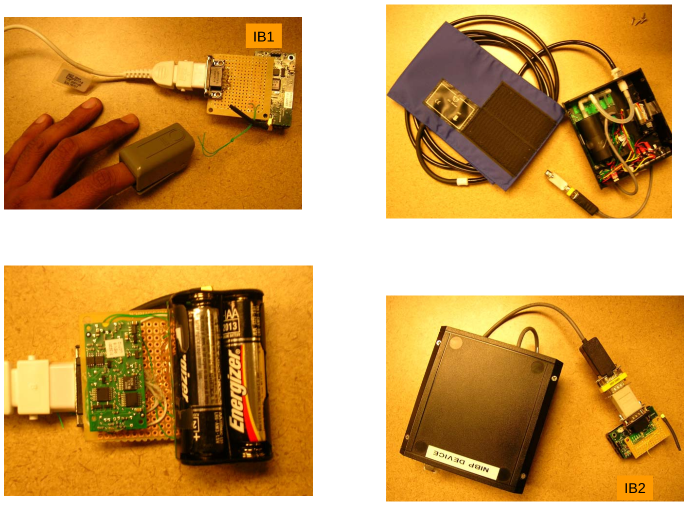


Fig. 25. Ayushman Implementation Pictures

Dell Axim PDA running Pocket PC 2002. The reason for choosing the PDA was because of its ubiquity, light-weight, processing capability, multi-model communication capability and screen for data visualization. As the PDA does not have the ZigBee interface, we connected a mote to the PDA both of which communicated using the serial interface. Our initial prototype was done using the Crossbow StarGate [1] node running embedded linux. Our current implementation allows using either devices as base stations. Our implementation of sensor network utilized multiple hardware and communication configurations, which shows Ayushman's technology independence. As long as the the required functionalities are provided and the data formats are maintained any technology can be used to collect and communicate patient data.

5.3.2. Software Components

Our implementation of Ayushman has a two-tier setup. At the first tier sensors collect patient data and forward it to the base station, which itself forwards the data to the central server. The data collection end of Ayushman is implemented using nesC and runs of TinyOS 1.1.15 for both MicaZ and TelosB motes [1]. The code for TelosB motes samples its in-built sensors every 5 seconds and sends the data out. For each patient, therefore, environmental data is collected frequently. The MicaZ motes collects physiological data at different rates and in different manner depending upon the type of the sensor/device and need. The accelerometer which is biaxial, and is part of the pluggable sensor board, is sampled every 5 seconds (based

TABLE 6

Sensors and Mode of Data Collection

Parameter	Interface	Mote	Interval	Mode
Blood Pressure	Serial	MicaZ	2 min	Pull
SPO2	Serial	MicaZ	2 seconds	Push
Acceleration	51 pin conn.	MicaZ	5 sec	Pull
Temperature, Humidity, Light	In built	TelosB	5 sec	Pull

on a timer) to obtain the current X and Y axes values. The blood pressure data is collected once every two minutes, by sending it an explicit command to the device which starts the measurement process. Table 6 summarizes the details. Each measurement by the BP monitoring device provides three physiological parameters: systolic/diastolic pressure, heart rate and mean arterial pressure. The SPO2 device on the other hand works in the push mode and once connected keeps sending data out approximately once every two seconds. It too provides the following physiological parameters: % of SPO2 (blood oxygenation), pulse rate, pulse strength, and plethysmogram with each measurement. The sensors and medical devices send a stream of raw data which need to be parsed and processed according to their specifications to obtain physiological parameters. This is done at the base station. In the sequel we discuss our implementation of the base station using a PDA. The other implementations are similar in functionality. The base station receives data from each mote in one hop, forming a star topology with the base station forming the hub. The base station has been implemented using C#. Other topologies and multi-hop networks can be easily implemented without loss of functionality. Once the patient data has been processed, the derived physiological parameters are sent over to a central server. The central sever stores the data received in an MS Access database and provides an IP based interface for remote clients. The data communication between the base station and the central server is done using XML format which allows platform independent way of communicating patient data. We implemented the central server to execute on a laptop running windows. The base station forms an ad-hoc WiFi connection with the central sever to communicate.

Remote clients connect to central server at a specific port to query patient data. The central server provides multiple ports for receiving queries from the patients, this allows remote clients to view both past

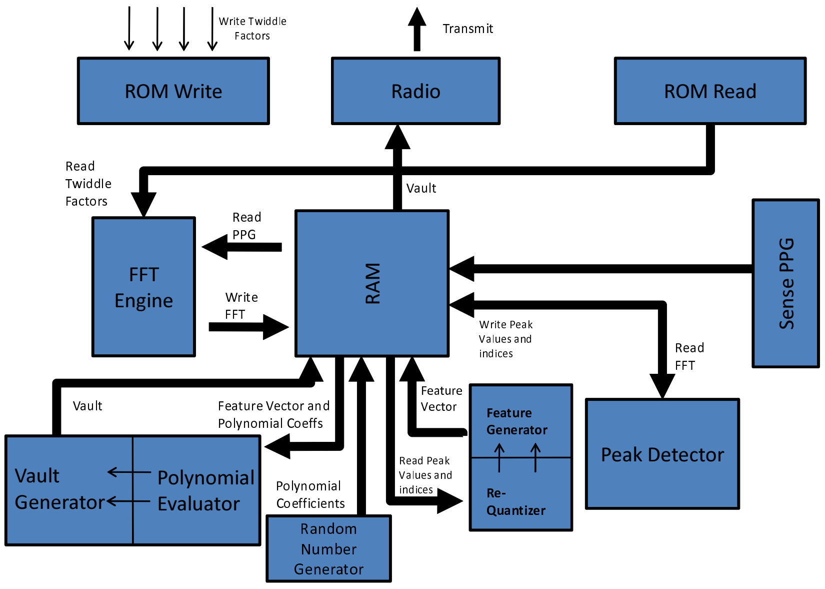


Fig. 26. Implementation of the Vault Generation Module (with FFT Module)

and real-time data from a patient at the same time. The client end also provides a means of receiving threshold based alarms. For each physiological and environmental data that a client receives, it can set maximum and minimum threshold values, if the received values is not within these limits an alarm is sounded. The client can on the fly change the thresholds to accommodate values which may not be construed as normal depending upon the way the patient is behaving.

Now that we have our testbed, we validated our protocols on this infrastructure. The PSKA protocol will be tested by using it to secure the communication between the physiological and environmental sensors. The CAAC access control model was executed at the central server which intercepts and verify every access attempt made by remote clients before providing them access to patient information according to the demands of the situation.

5.4. Implementation of PSKA on Ayushman

In this section we present the details of implementing PSKA on the Crossbow motes platform using TinyOS as a part of the Aysuhman PHMS. We begin by describing the implementation itself, followed by a look at how it performed with respect to our Matlab benchmark. Our implementation of PSKA utilized two

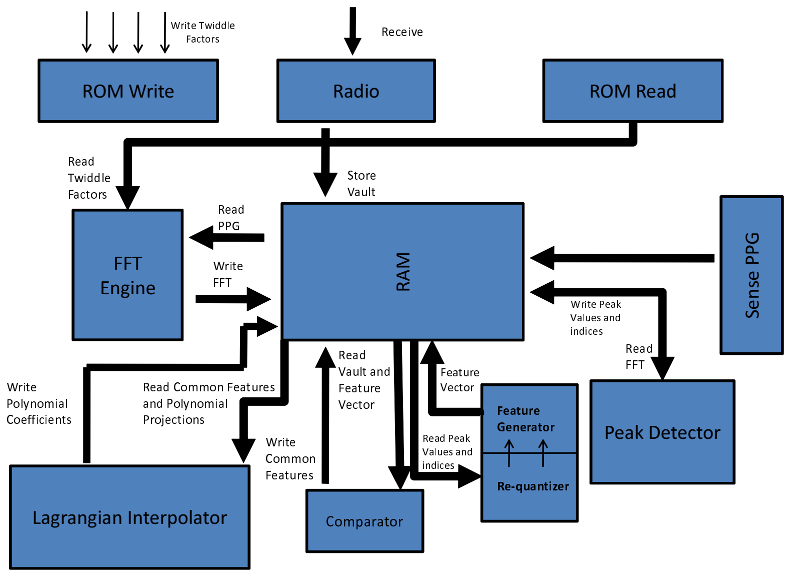


Fig. 27. Implementation of Vault Unlock Module (with FFT Module)

Telos motes [1]. We used PPG signal as physiological signal of choice for our implementation as we had the hardware to measure it. PSKA essentially secures the data gathering aspect of the PHMS conceptual model.

5.4.1. Implementation Modules

PSKA is a very memory and communication intensive protocol. The implementation of the PSKA system was done by dividing its functionality into 3 modules - *FFT Generation Module*, *Vault Generation Module* and *Vault Unlock Module*. In order to execute one run of PSKA, the FFT Generation Module is first executed at both the sender and receiver motes. At the sender the FFT coefficients thus obtained are provided for the Vault Generation Module, which was executed next. The Vault Generation Module generated the peaks from the FFT coefficients and creates the vault. The vault was then transmitted to the receiver, where the FFT coefficients have been similarly provided to the Vault Unlock Module which then performs the unlocking operation and decodes the secret key.

FFT Generation Module: The FFT was implemented using the Cooley-Tukey algorithm which requires O(N$log_2 N$) complex multiplications where N is the number of FFT points, which in this case was 256. Typically, FFT computations require floating point arithmetic operations, which are not available in the mote platform. Therefore, in order to implement FFT on motes, all floating point numbers were represented using a fixed point system, rather than using the complex IEEE 754 standard. This was done in a three step process: 1) First, the floating point numbers are represented in binary-point format; 2) The binary-point representation is shifted seven places to the right and all the bits to right of the binary-point are ignored; 3) The bits to the left of the binary point are then used to represent the floating point number in the 32 bit fixed point representation. The principal idea behind the process is to multiply the floating point number by 128 and to store the integer portion of the result. This effectively provides a representation of the floating point number with a resolution of two decimal places. Even though we lose some of the precision in the process, this representation did not affect the accuracy of FFT calculation. Figure 28 shows the comparison between the FFT calculated by the motes and FFT computed using our Matlab benchmark, for one of the volunteers. The root mean square error (RMS) between the two calculated as a percentage of the mean value is about 1% which is enough for our purposes. Now that FFT can be computed on the motes, the next step is to derive features from the PPG signal. In order to do so, the measured PPG signal time-series was first divided into overlapping windows and FFT was computed on each of these windows. The first 32 FFT coefficients from each window were then concatenated to generate features [9].

Vault Generation Module: The feature extraction process involved the detection of peaks in the FFT coefficients. The peak detection algorithm simply scans through the FFT coefficients and tries to find the local maxima. The peak indexes and the associated FFT coefficient values at those peak indexes (peak values) are quantized exponentially to form a *feature point* (or feature value), which are then combined over the entire FFT to form the feature vector. Each feature point in the resulting vector is 32 bits long, as opposed to the 13 bit in the Matlab implementation, one of the down sides of simplifying the numerical representation. Once the features were derived at both the sensors, the key agreement process began which uses the fuzzy vault construct [54]. Therefore, one of the two motes (the sender) engaged in the key agreement, first selects polynomial order, say v. It then generates $v + 1$ random numbers using the random number generator that is available in TinyOS and is based on the Mid-Square method. Each of the $v + 1$ random numbers form the coefficients of the vth order polynomial. These random coefficients form the basis of the key to be agreed between the sensors, which can be obtained by simply concatenating the coefficients together. The choice of

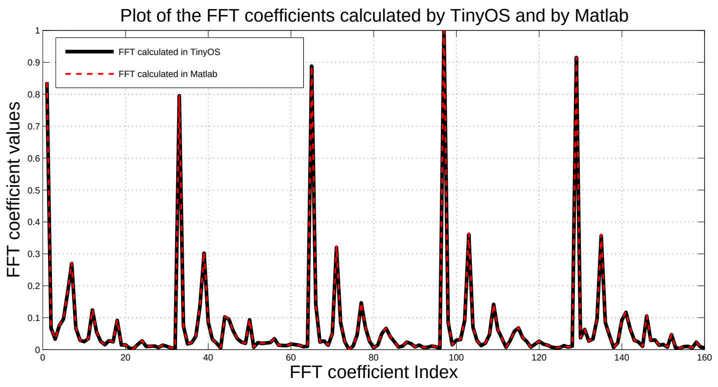


Fig. 28. FFT Coefficients Comparison between TinyOS- and Benchmark Implementation for PPG Data

v is is based on the common feature between same and different subjects as discussed in Section 3.6. The polynomial is then evaluated at each point in the feature vector. Each feature value and its projection on the polynomial are combined to form a tuple, which is then mixed with chaff (random) tuples whose values are in the same range as the legitimate tuples but do not fall on the polynomial. This Fuzzy Vault is then sent to the receiver mote. Figure 26 shows the block diagram of TinyOS implementation of the Vault Generation Module.

Vault Unlock Module: The receiver mote compares each of the feature point part of the tuples in the Fuzzy Vault with the feature points in the local feature vector. It isolates $v + 1$ such tuples and then uses Lagrangian interpolation to attempt to derive the polynomial (as thus the key). A comparator was implemented in TinyOS that simply scans through the vault for common feature points and stops whenever $v + 1$ common feature points are obtained. For performing polynomial interpolation a Lagrangian interpolator was implemented. Figure 27 shows the block diagram of the TinyOS implementation of the Vault Unlock Module.

5.4.2. Comparison with Benchmark

In order to validate our implementation, we executed the TinyOS implementation of PSKA with the same data set used for the Matlab implementation. Table 7 summarizes the measurement parameters for our implementation of PSKA and its comparison with the "ideal" implementation on Matlab as presented in [127] [125]. It can be seen that the number of average number of common features differ slightly from the values we obtained from the Matlab benchmark. The average feature length which is the number of

TABLE 7

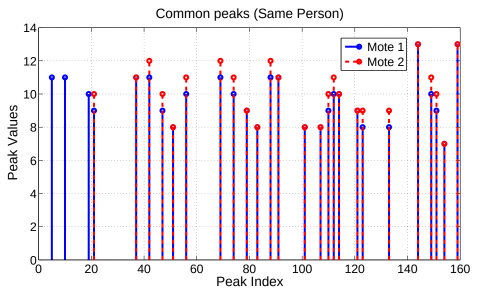
TinyOS-Implementation: PSKA Feature Generation Parameters and Statistics

Parameters	**Matlab**	**TinyOS**
Sampling	60 Hz	-Same -
Sampling Duration	12.8 secs	-Same -
FFT	256 points, 5 windows[a]	-Same -
Avg. Feature Vector Length	∼30	∼25
Avg. # Common Features (Same Host)	12	9.7
Avg. # Common Features (Different Hosts)	2	1.31
Feature Length	13 bits	32 bits
Polynomial Order Used	9	6

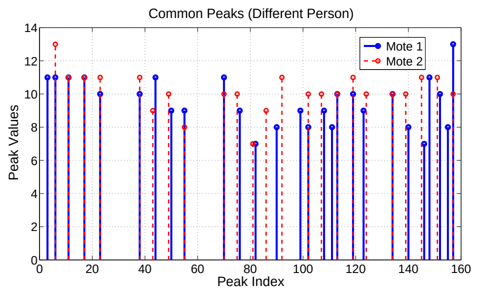
[a]First 32 points/window were concatenated together for peak based feature generation

peaks observed in the FFT of the PPG signals is about 30 in the benchmark and 25 in TinyOS. Further, the average number of common features obtained at the two sensors on the same host is 9 compared to 12 in the benchmark, while the value between the common features between two different subjects is 1.31 compared to around 2 in the benchmark. The primary reason for this difference is the loss of resolution in the representing the floating point numbers as discussed above. However the substantial gap between the number of common features between the same and different subjects shows that our implementation does not suffer due to this approximation. Figure 29 shows an example of the number of common features (peak values and peak indexes) between same and different subjects. By setting the polynomial order to a value between the number of common features for the same and different subjects we can ensure the safety of the vault during transfer from the sender to the receiver. Further, the difference in the common features also provides a form of authentication for the communicating sensors. If the receiver is able to unlock the vault, then it knows the vault was sent from another mote on the same host. This is an important property of key agreement using physiological signals which other traditional key agreement schemes such as Diffie-Hellman do not possess.

A significant difference between the benchmark Matlab implementation of PPG-based PSKA and the TinyOS version was the order of the polynomial used. In our original implementation we used the 9th order polynomial which we found to provide a good balance in reducing the *false positives* - finding the number of common features generated by sensors with data from two different subjects greater than $v + 1$ and *false negatives* - finding the number of common features generated by sensors with data from the same subjects less than $v + 1$, and providing security against brute-forcing the polynomial reconstruction. In the case of TinyOS,



(a)

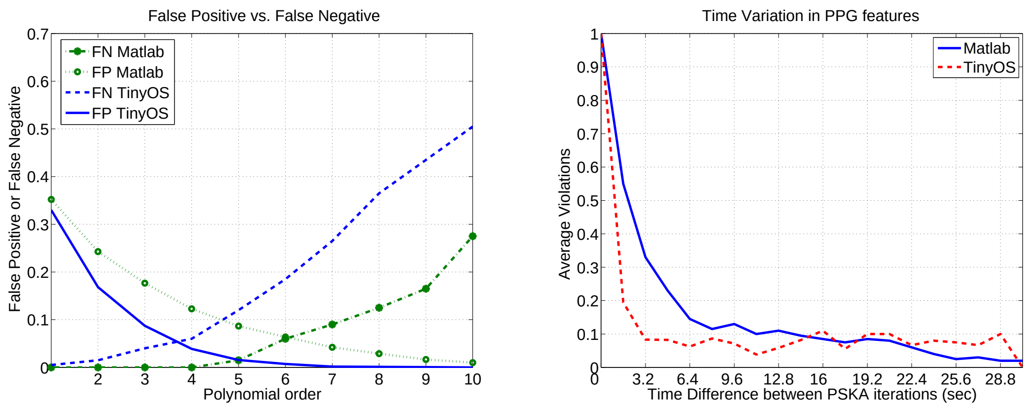


(b)

Fig. 29. TinyOS Implementation - Common Features (FFT peak index vs. FFT peak values) for: a) PPG signal measured by sensors on the same host (Total = 12 common features), b) PPG signal measured by sensors on the two different hosts (Total = 3 common features)

we had to limit the polynomial order to 6 because of overflow problems we faced during the projection of the feature points on the polynomial during vault creation. Figure 30 (a) shows the false positives and false negative for TinyOS implementation and the benchmark. The results for TinyOS are extrapolated to 10th order polynomial. We find that as the TinyOS implementation has on average better false positive rate than the benchmark. This is mainly because of the TinyOS implementation has lower number of total features than the benchmark and consequently lower number of common features between two different subjects. By the same account, the false negative rate performance of TinyOS implementation is poorer than the benchmark as for the same person the number of common features may fall below the order of polynomial used. This shows that TinyOS implementation maintains the distinctive property of PSKA and can enable authenticated key agreement. Figure 30 (b) shows the time variance in the PPG signal features (when 6th order polynomial is used) measured as *average violation* - which is the percentage of times when the number of common features between two iterations of PSKA are greater than the order of the polynomial used. We can see that as the time between two iterations increases the violations falls dramatically as in the case of the benchmark. These results show the *accuracy* of the TinyOS implementation of PSKA thus meeting our implementation challenge.

Another important aspect of the PSKA protocol is the chaff point generation and the transfer of the vault. As we are using only a 6th order polynomial, in order to reduce the chances of brute-forcing the vault we add



(a) (b)

Fig. 30. Mote Implementation - False Positive and False Negative Comparison for TinyOS implementation of PSKA with Benchmark, b)Time Variance Comparison for TinyOS implementation of PSKA with Benchmark

a large number of chaff-points. Our implementation of PSKA involved 1000 chaff-points, which are tuples of 32 bit random numbers which are generated as required. Therefore, when the vault has to be transferred to the receiver, the chaff points are generated in groups of 10 and stored in the RAM. They are then mixed with subset of points in the feature vector and transmitted to the receiver over multiple TinyOS packets. For the transmission of the vault, the sender mote split the vault down into chunks of 80 bytes each and sent it to the receiver mote. Details of the resource requirement and execution time for the PSKA protocol is provided in Section 5.4.3.

5.4.3. Usability Analysis

Now that we have presented our implementation of PSKA on TinyOS, in this section we focus on their performance aspects. The primary metrics for evaluating the two schemes are based on their defining characteristics - *usability*. We define usability in terms of our ability to implement the protocol on motes in order to achieve plug-n-play, transparent key agreement. The plug-n-play nature of PSKA is obvious, as the sensors simply have to measure PPG signals in order to be able to execute it. Therefore our focus for evaluating PSKA is to show that it can be implemented on the mote platform. We demonstrate this by computing - 1) the time taken to execute the PSKA scheme at the sender and receiver, 2) the memory used in the process, and 3) the robustness - ability of receiver mote to unlock the fuzzy vault in event of packet loss

TABLE 8

PSKA TinyOS Implementation Performance Results

Module	**RAM (Bytes)**	**Code Size (Bytes)**	**Time (msec)**
FFT (at each mote)	4690	17312	1080
Vault Generation	1750	15462	10089
Vault Unlock	1778	16902	106

during the vault transfer. The Table 5.4.3.1 summarize the performance results in terms of RAM usage, ROM usage and time taken by PSKA in our implementation.

5.4.3.1. Memory Requirements

The FFT computation is the most RAM intensive operation in the entire process. The reason for this is that a considerable amount of floating point multiplications and complex number operations are done. The Vault Generation module takes a considerably lower amount of RAM, as it only has to do peak detection, quantization and polynomial evaluation. Most of these operations are comparison based hence does not require as much RAM as the FFT computation. Vault Unlock module requires slightly higher RAM than the Vault Generation, primarily due to its need to perform Lagrangian interpolation which is more complex than polynomial evaluation. The code size for each of the module reflects their respective complexity. The FFT module being the most complex operation has a largest code size, followed by the Vault Unlock module due to the need for implementing the Lagrangian interpolation followed by the Vault Generation module whose implementation code is dominated by the polynomial generation.

5.4.3.2. Time Requirements

FFT computation phase includes five computation of 256 point FFT hence takes a greater amount of time than the Vault Generation and the Vault Unlocking module. The Vault Generation process is most time consuming of all the steps. The primary reason for this is the generation of chaff points and their transmission. For 1000 chaff points, the number of TinyOS packets sent is about 100, 80 byte packets. The time taken to send these packets to the receiver end is about 10058 msec. On the other hand, it takes only 31 msec to perform the other tasks of this module - peak detection, quantization and polynomial evaluation operations. The time consumption for the Vault Unlock module is dominated the Lagrangian interpolation. The Lagrangian inter-

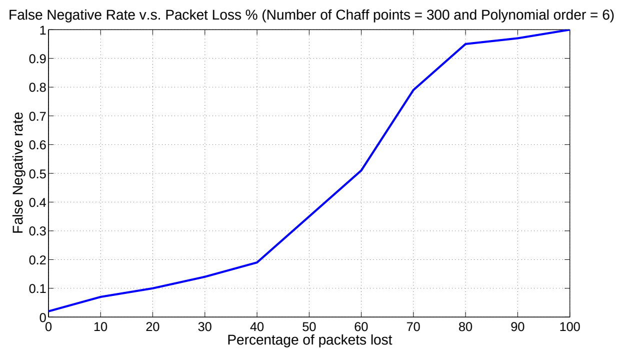


Fig. 31. Robustness of Mote-Implementation of PSKA

polator requires the computation of O($PolynomialOrder^2$) of polynomial multiplications. Each polynomial multiplications involves a convolution operation which again requires O($PolynomialOrder^2$) of multiplications. All the other aspects of the Vault Unlocking - peak detection and quantization is identical to Vault Generation module.

5.4.3.3. Robustness

One final property that we looked into was that of robustness of the PSKA protocol. As the exchange of vault is spread over many packets, we wanted to study what happens if the packets are lost. Figure 31 shows the percentage false negative rate for PPG-based PSKA with respect to packet loss - a metric we call *robustness*. It can be seen that as the % of packet loss goes up the false negative also increases, mainly because, as the packets get lost, the number of common feature points in the feature vectors (at the sender and receiver) also decrease. This reduces the chances of the successful vault unlocking. However, we see that PSKA protocol is quite robust, as even with the packet loss rate of 40% the false negative still remains relatively low.

From these results we argue that PSKA is not only plug-n-play but can also be implemented on mote platform easily and is highly robust- demonstrating its usability. From these results we can see that, the usability provided by PSKA comes at a cost of higher memory and communication cost from the sensors. A direct consequence of using PSKA is the greater energy consumption for each node performing key agreement. This energy cost would be completely avoided in traditional pre-deployment based approaches. However,

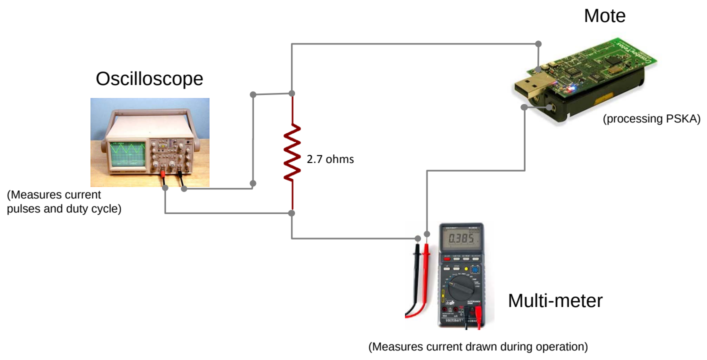

Mote
Oscilloscope
(processing PSKA)
2.7 ohms
(Measures current pulses and duty cycle)
Multi-meter
(Measures current drawn during operation)

Fig. 32. Energy Measurement Setup

we argue that with an ever increasing memory and battery capabilities, the overheads imposed by PSKA on motes can be easily overcome.

5.4.4. Energy Analysis

Given the cyber-physical nature of PSKA, it requires many signal processing and mathematical routines in order to function, which can be very expensive for the motes. It is therefore important to characterize the energy consumption (footprint) of PSKA. We analyze the overall energy footprint of PSKA to see what aspects of the PSKA protocol are the most expensive. We also look at the computation and communication energy costs for PSKA, which we believe are comparable; the former therefore cannot be ignored unlike many prominent non-cyber physical solutions such as [90] [142]. We further investigate prominent energy scavenging techniques to see if they can meet PSKA's energy requirements. The principal idea being to evaluate the possibility of eliminating the energy cost of utilizing PSKA for the sensors, within the BAN. In a way, by showing that energy scavenging can be used to meet PSKA's energy requirements, we are extending its plug-n-play nature from a solely security perspective to its operation as well (the operation of PSKA does not require any maintenance by the BAN user, deployment alone is sufficient). In order to compute and analyze the energy consumption of PSKA, we first define the energy consumption model used by us. We then compute the energy consumed by different stages of PSKA and finally analyze them in detail.

5.4.4.1. Energy Model

There are two distinct components to energy consumption in PSKA that we need to consider. The first is the *computational energy cost*, which quantifies the amount of energy consumed during the execution of PSKA, while the second is the communication energy, which quantifies the energy consumed in transmitting and receiving the vault. Traditionally, in the domain of sensor networks, the prevailing assumption has been that *communication energy costs* overwhelm computational costs. However due to the considerable processing requirement of PSKA in terms of FFT computation, feature generation, polynomial generation, evaluation and Lagrangian interpolation, we suspect computational costs to be substantial portion of the total energy costs.

Computational Energy Consumption Model: Our energy model is based on the on-line energy estimation model described in [36]. The idea behind the model is to determine the time for which each hardware component in the system is on or off, and the current it draws during the process (functioning and idle). Energy consumed can then be determined by multiplying the current and time with the supply voltage (V). More formally, the model can be represented using the following linear equation:

$$E_{comp} = V \times (I_{pr}t_{pr} + I_{ps}t_{ps} + \sum_{i} I_{c_i}t_{c_i}) \tag{5.1}$$

Here, E_{comp} is the computational energy cost, V is the voltage used by the system, I_{pr} and t_{pr} are the current draw when the processor is running and the time for which it is running, respectively, I_{ps} and t_{ps} are the current draw when the processor is in the idle mode and the time for which it is in that mode, respectively, and $\sum_i I_{c_i}t_{c_i}$ gives the current and time required for the other components on the mote, for example sensing.

Communication Energy Consumption Model: The second important aspect is the communication energy requirements of PSKA. We again use a model similar to the one computational one here by determining the time spent in transmission of the packets and time current drawn in process, and time spent in the receiver mode and its corresponding current draw. More formally:

TABLE 9

TinyOS PSKA Implementation Stages' Current Draw and Timing Results

Mote	Stage	Current Draw (Radio-Off)	Current Draw (Radio-On)	Time (msec)
Sender/Receiver	Sensing	6.6mA	6.6mA	12700
	FFT Computation	1mA	19.56mA	2138
	Peak Detection and Quantiz.	0.14mA	18.72mA	12.4
	Feature Generation	0.11mA	18.72mA	13.6
Sender	Polynomial Gen. & Eval.	0.08mA	18.68mA	8
	Chaff Points Gen.	0.01mA	18.61mA	14 per 10 points
	Vault Tx	-	19.33mA	1350(1K), 2700(2K), 4000(3K), 5360(4K), 6750(5K)
	Ack Rx	-	19.20mA	20
Receiver	Vault Rx	-	19.41mA	1400(1K), 2750(2K), 4100(3K), 5370(4K), 6760(5K)
	Lagrangian Interpolation	0.43mA	19.04mA	50
	Ack Tx	-	19.11mA	17

$$E_{comm} = V \times (I_{tx}t_{tx} + I_{rx}t_{rx} + I_{roff}t_{roff}) \tag{5.2}$$

Here, E_{comm} is the computational energy cost, I_{tx} and I_{rx} are the current draw by the transceiver during transmission and reception, respectively, t_{tx} and t_{rx} is the time taken for transmission and reception respectively. While I_{roff} and t_{roff} are the current draw when the transceiver is switched off and the duration for which it is off, respectively, and V is the supply voltage.

5.4.4.2. Energy Consumption of PSKA

In this section, we present the result and analysis of the energy consumption of PSKA protocol. Our results were through instrumentation of an implementation of PSKA, using which we directly measured the current drawn for different stages of PSKA and their duration of execution. Our aim was to determine the energy consumed by various stages of PSKA for *one complete iteration of PSKA*. We begin by describing our experimental setup and our results and then move on to analyzing them.

Results: The aim of the experiments was to determine the current drawn and its duration in different stages of

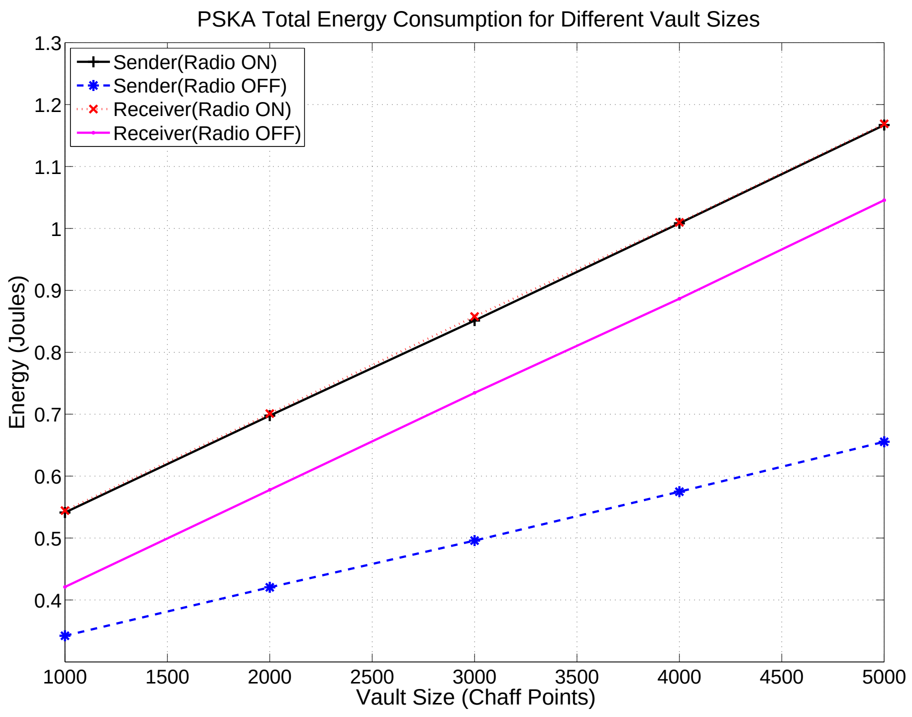


Fig. 33. TinyOS Implementation of PSKA: Total Energy Consumption for different Vault Sizes

the PSKA execution so as to determine the energy consumed based on our energy models. In order to do so, we established an experimental setup as shown in Figure 32. As mentioned earlier, the protocol was executed on a pair of TelosB motes which attempted to use PPG signal from the human body in order to agree on a common key. Across the two power leads of the mote, a small resistance (2.7 ohm) was connected in series with an ammeter. An oscilloscope was connected across the resistance so as to visualize the voltage pulses generated (across the resistor) and to measure their duty cycle. The resistance value was chosen such that the voltage generated across it by the current pulse is clearly visible in the oscilloscope. Care was taken that the resistance chosen is not too large to affect the input impedance of the mote (which acts as a current source in this case).

In order to be able to analyze the energy consumption of the PSKA protocol, our energy measurements divided a single execution of the protocol into eleven stages: 1) Sensing 2) FFT Computation, 3) Peak Detection and Quantization, 4) Feature Generation, 5) Polynomial Generation and Evaluation, 6) Chaff Points

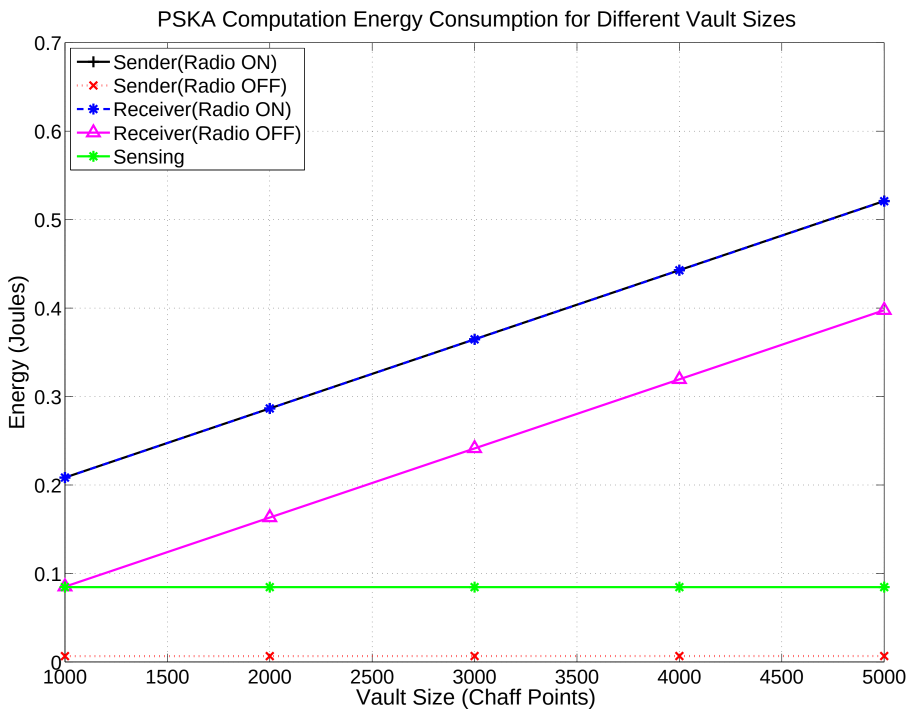


Fig. 34. TinyOS Implementation of PSKA: Computation Energy Consumption for different Vault Sizes

Generation, 7) Transmission of Vault, 8) Reception of Vault, 9) Lagrangian Interpolation, 10) Transmission of Ack, and 11) Reception of Ack. Stages 2 through 4 were executed by both the sender and receiver motes. Stages 5–7 and 11 were executed only by the sender, while Stages 8–10 were executed by the receiver only. We performed two sets of experiments, one with radio on during the entire execution of the PSKA, in order to see the worst-case performance of the protocol. In the other experiment, the radio was switched on only when needed, i.e. when the sender and receiver are expecting to receive a message or transmit a packet (Stages 7–11 for the sender, and Stages 7–9 and 10 for the receiver). For brevity, in the rest of the section, we refer to the former set of experiments as *Radio-On* and latter as *Radio-Off*. Table 9 shows the results obtained from our experiments. The values closely match those reported for different functions of a TelosB mote in its original evaluation [92]. It can be seen from these results that the FFT computation, vault communication and acknowledgment, and Lagrangian interpolation are the most expensive stages in terms of computation. When the processor is in the idle state, we found that the mote consumed 0.01mA when radio was off, and 18.60mA

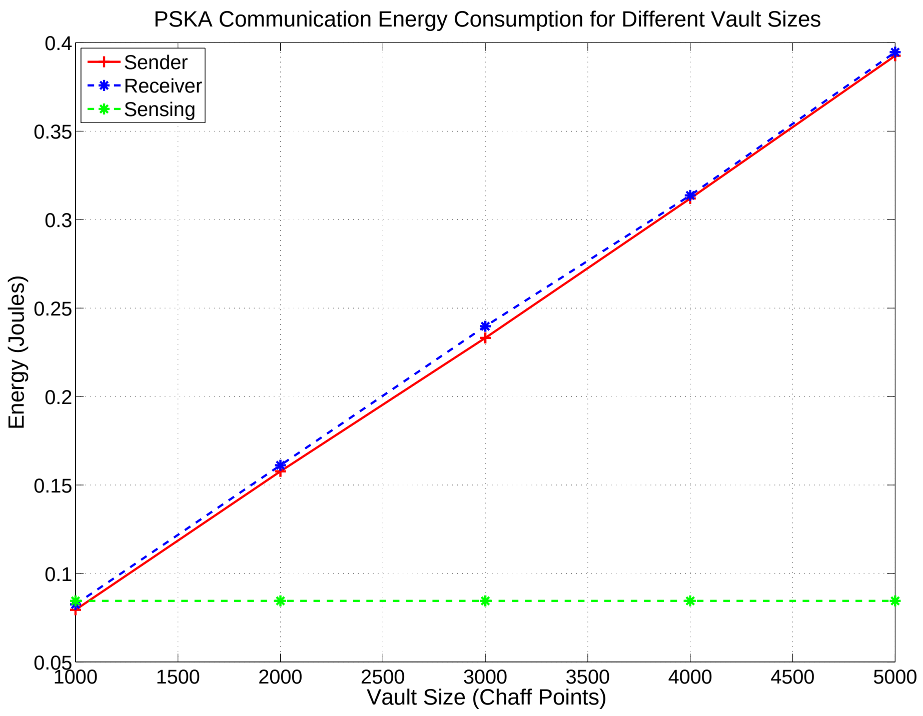


Fig. 35. TinyOS Implementation of PSKA: Communication Energy Consumption for different Vault Sizes

when the radio was on. It can be seen that the current draw for the Radio-On experiment goes up dramatically for the mote. Now that we have the current draw results, we can now analyze the energy consumption of the entire protocol.

Analysis: The analysis of the energy consumption results has two parts: analyzing the energy costs for PSKA, and analyzing the computation and communication energy costs for the protocol. The primary source of security for PSKA for a given polynomial order is the number of elements in the vault. The larger the vault, the greater the number of combinations an attacker has to try to arrive at the correct key [127] [125]. We therefore compare the energy consumed by one complete iteration of PSKA with respect to different chaff points in the vault.

Figure 33 shows the overall energy consumption by the sender and receiver for executing the protocol at five different vault sizes. The energy consumed in creating and opening larger vaults is greater than small vaults. This is because smaller vaults have lesser number of chaff points which sender needs to add to the

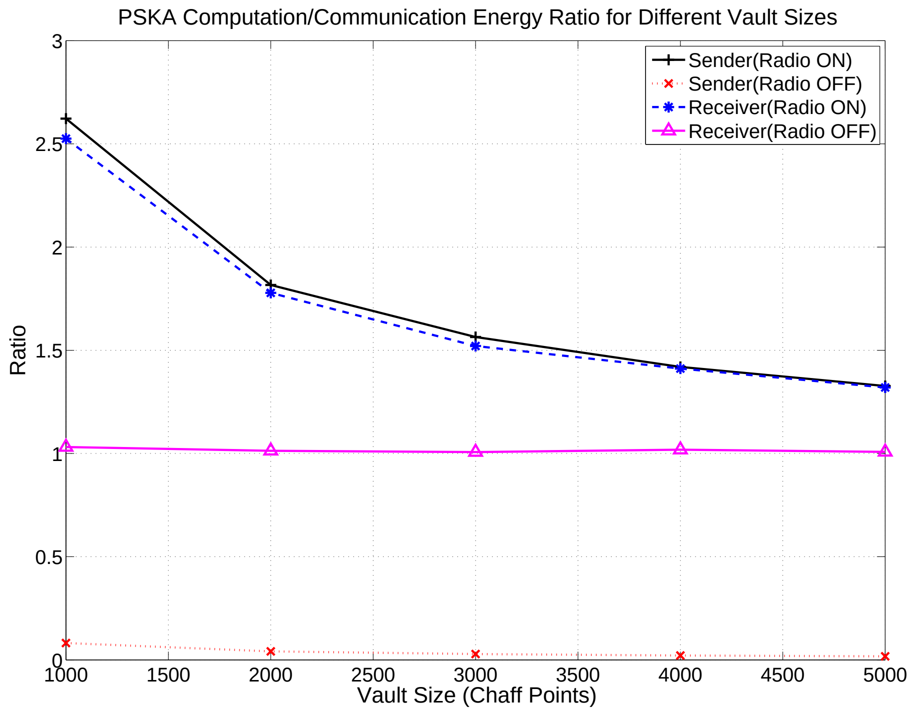


Fig. 36. TinyOS Implementation of PSKA: Computation and Communication Energy Ratio for different Vault Sizes

legitimate points. Consequently lesser the number of packets need to be transmitted in order to communicate the vault to the receiver. Similarly, at the receiver's end, the number of combinations of the vault elements that the receiver has to compare with its own feature points, to identify the polynomial, is much smaller along with the time taken to receive the vault itself. Further, in the Radio-On experiment the receiver consumes slightly more energy than the sender as more current is drawn in receiving the packets than sending them. In the Radio-Off experiment, the sender consumes much less energy because it gets to remain off longer. These results underline the traditional *trade-off between security and energy*.

Figure 34 shows the computational energy consumption, for both sender and receiver during both Radio-Off and Radio-Off experiments with respect to the cost of sensing the PPG signal (which is a constant value). As expected, the cost of computing the larger vaults causes the energy consumption for the mote to go up as the vault size increases. Similarly, Figure 35shows the communication energy consumption with respect to sensing cost, for both sender and receiver. It shows an increase in energy cost as the vault size increases due to the increased number of packets that need to be transmitted from the sender to the receiver. In both cases, for Radio-Off experiment, the sender consumes much less energy than all others because it gets to remain off longer.

We then compared the percentage of energy cost that computation takes compared to communication. Figure 36 shows the results of our comparison which plots the ratio of computation cost and communication cost for different vault sizes. We find that for smaller vault sizes, the cost of computation outweighs communication by a factor of almost two (for Radio-On). But for larger vault sizes, which require extensive communication, the associated cost pre-dominates. This shows that computation energy cost for PSKA is comparable to the communication energy cost, and should not be ignored. For Radio-Off experiments, the cost of computation is comparable to communication for receiver, and much lower for the sender.

The energy consumption for PSKA essentially describes the cost of having a plug-n-play key agreement protocol. Traditional key agreement protocols either utilize key pre-deployment [90] or public key cryptography [74]. Key pre-deployment is free as it based on manual initialization, while Elliptic Curve Cryptosystem Diffie-Hellman (DH-ECC) protocol distributing a 163 bit key consumes about, 0.8J on the mote platform [74]. This illustrates the other important trade-off when it comes to PSKA: *trade-off between usability and energy*. Highly usable solutions also have a greater cost, and this property needs to be considered when deploying BANs.

5.4.4.3. Energy Scavenging

Now that we know the energy requirements of PSKA, in this section we present a semi-formal discussion on the possibility of utilizing energy scavenging techniques in order to meet its (PSKA's) requirements. The reason for investigating the use of energy scavenging techniques for "powering" PSKA is to investigate if it can be made self-sufficient in terms of energy. Again, we focus on evaluating the scavenging techniques for powering the mote for one complete iteration of PSKA. We believe this ability to use energy scavenging to meet its requirements, enhances the plug-n-play nature of PSKA by making it transparent to its users (host of the BAN) in terms of its energy requirements as well.

TABLE 10

Energy Scavenging Techniques for PSKA

Scavenging Techniques	**Source**	**Power Gain**	**Ideal Deployment Scenario**
Body Heat	Latent heat of vaporization of perspiration	0.2W - 0.32W	Most Cases
Respiration	Chest expansion from breathing	420mW	Strenuous Physical Activity
Ambulation	Arm & leg movement	1.5W-1.6W	Physical Movement
Photovoltaic Cells	Sunlight	$100mW/cm^2$	Exposure to sun

Currently, the TelosB motes that are being used to implement PSKA in the BAN use AA batteries, from which they draw power for their operation. A typical AA battery has a voltage rating of 1.5 V and is rated at 2600 mAHour at 19 mA current (`http://data.energizer.com/PDFs/E91.pdf`). In the worst case (Radio-On scenario[1] at vault size of 5000 for the receiver), PSKA requires an average power of 40.6mW (1.1689J energy for 28.8 seconds). The maximum power consumed by PSKA is 58.8mW during the FFT computation. Thus with two AA batteries, used in the motes, we can run more than 6000 iterations of PSKA. This is not a surprising result. To put things in perspective, the cost of normal operation of a mote (assuming it requires sensing, processing and transmission of a 30 byte data every n seconds) roughly consumes a maximum power of 57mW (during data communication). PSKA versions with larger vault sizes or polynomial orders will be much more expensive. However, if we do not execute PSKA regularly the overall effect of PSKA on the mote can be lowered.

However, our aim here is to make PSKA transparent to its users with respect to its energy requirements. We therefore look at some of the prominent energy scavenging techniques that can balance the energy requirement of PSKA so that the motes battery be "spared", allowing it to be used for the regular sensing processing storing and transmitting operations. Much work has been done in energy scavenging in the domain for wearable sensors [99] [140] [141]. The focus of these works is mainly on developing hardware for scavenging. In [85] [119] the authors present a comprehensive study of the amount of energy available to be scavenged from different sources on the human body in for operating mobile devices.

In this work we discuss four prominent sources from which we can scavenge energy for running PSKA.

[1]We only consider the case of Radio On in this analysis as meet its energy requirement is guaranteed to meet the requirement of the Radio Off scenario.

Note that we assume an *on-the-fly energy provisioning model* in which the energy is consumed as it is generated.

- **Body Heat:** Human body heat is a source of energy that can be scavenged. In [85], the authors suggest the use of neck braces that can be worn by host of the BAN to enable the scavenging. These neck braces scavenge energy from the latent heat of vaporization occurring due to the vaporization of the perspiration of the individual. The estimated power gain is 0.2 W to 0.32 W with this approach, which is sufficient for PSKA.
- **Respiration:** Any form of movement on the body can be used to scavenge energy [85]. One of the prominent movement is due to respiration. It is estimated that about 420 mW of power can be extracted from a stretchable dielectric elastic band worn around the chest of an individual. This amount of power is sufficient for PSKA, but can be extracted only from heavy breathing.
- **Ambulation:** Demonstrated systems have been successful in recovering 1.5 W of power from the human arm motion and more than 1.6 W of power from the ambulatory motion of human beings (using piezo-electric soles in shoes) [85], which is sufficient for PSKA.
- **Photovoltaic Cells:** Photovoltaic cells can produce 100 mW/cm^2 of power when under sunlight [85], which is again sufficient for PSKA.

It can be seen that PSKA in the worst-case is energy-efficient enough to be sustained by each one of energy scavenging techniques described above. Further, eliminating the need for batteries also improves the overall green-nature of the scheme. Other sources such as those which depend upon vibration, blood pressure or from radio transmission have also been developed [119], but they do not provide enough energy to meet the requirements of a single iteration of PSKA. It can be seen from the energy characterization of PSKA that these techniques are sufficient to sustain the PSKA operation in a mote. The choice of a particular technique depends upon the scenario of deployment. For example, in the case of a patient who is completely bed-ridden one cannot use the energy scavenging techniques that are dependent on ambulation or on heavy breathing.

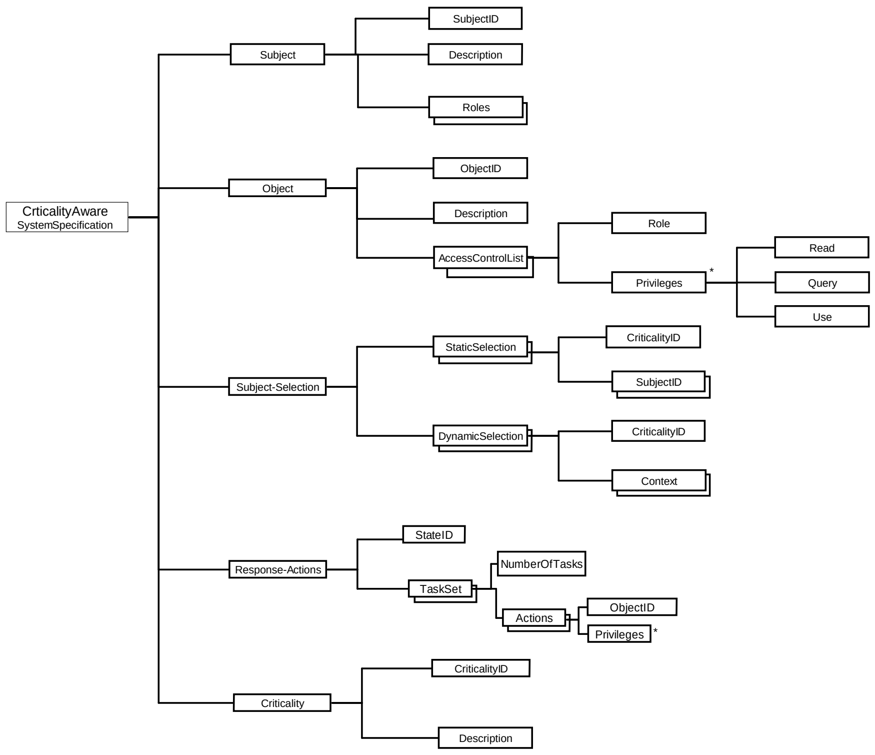


Fig. 37. Schema for Criticality Description Language

Other techniques such as photovoltaic cells or body heat have to be used. However for an athlete the energy scavenging system can make use of the physical movements or the heavy breathing to generate enough energy for executing PSKA. Table 10 summarizes the results.

5.5. Implementation of CAAC on Ayushman

Once the patient's medical data is collected by the sensors, it is forwarded to more capable entities for storage, process, and visualization. However, the primary task of a PHMS is to disseminate the data to clients such as caregivers, in order to facilitate the continuous monitoring of patients. But a patient's information cannot be released to anyone (due to the sensitive nature of the data), care has to be taken that only authorized access is possible. Further, by definition pervasive health monitoring systems enable patients to go about their lives while being continuously monitored. The idea being that, the medical monitoring infrastructure can enable the detection of onset of medical problems in time. Once such a problem is detected, the PHMS should also provide the ability to convey this information to appropriate entities (e.g. first responders) for facilitating patient care.

The CAAC was implemented as a part of central sever of the Ayushman PHMS. The CAAC policy specifications are specified using a specific XML based language called Criticality Description Language (CDL). The schema of CDL is given in Figure 37. The system administrator can specify both the normal operation and criticality management policies using this schema. Any access attempt by a subject is evaluated based on the policies specified. For example, in a PHMS, if the subject requests for blood pressure data from a patient X, it is provided only if the read privilege on the appropriate object evaluates to true. Any aspect of the schema can be easily extended to incorporate more complex policy requirements of the system it is deployed in.

Figure 38 shows the implementation of CAAC specifications. The central component of the system is CAAC coordinator which manages the entire process. The *Policy Manager* stores the CDL based policy for criticality handling. The modules on the left are principally used for criticality management. The *Subject Manager* manages subjects in the system, their roles and their current (active) role in the system. The *Object Manager* similarly manages the objects in the system and their ACLs. The Subject *Selection Manager* and the *Action Generation Manager* utilize their respective rules for determining the actions (objects and privileges) and subjects for a given criticality. The modules on the right are used to perform access control and auditing for accountability purposes. The former checks to see if the privileges demanded by the client match with the ones available to their role on the object being accessed. The latter keeps track of the current state of PHMS including information on - time the alternate privileges that were assigned at any given time, the subjects to which they were assigned and the time for which they were assigned, current state of the system, subjects, and objects. The critical events manager component is responsible for detecting the state of the overall system to determine the presence of the criticality, which then initiates the criticality management.

Figure 39 illustrates a sample operation of CAAC on the Ayushman PHMS, for data from a single patient. We are limited in the scope of our example by the availability of probability values for the ailments we consider and their respective response actions. We consider data collected from four data streams from a

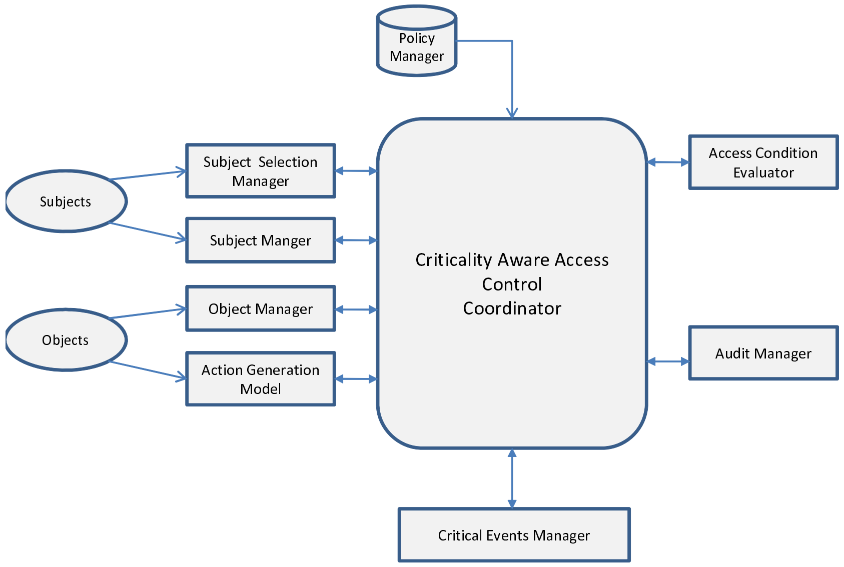
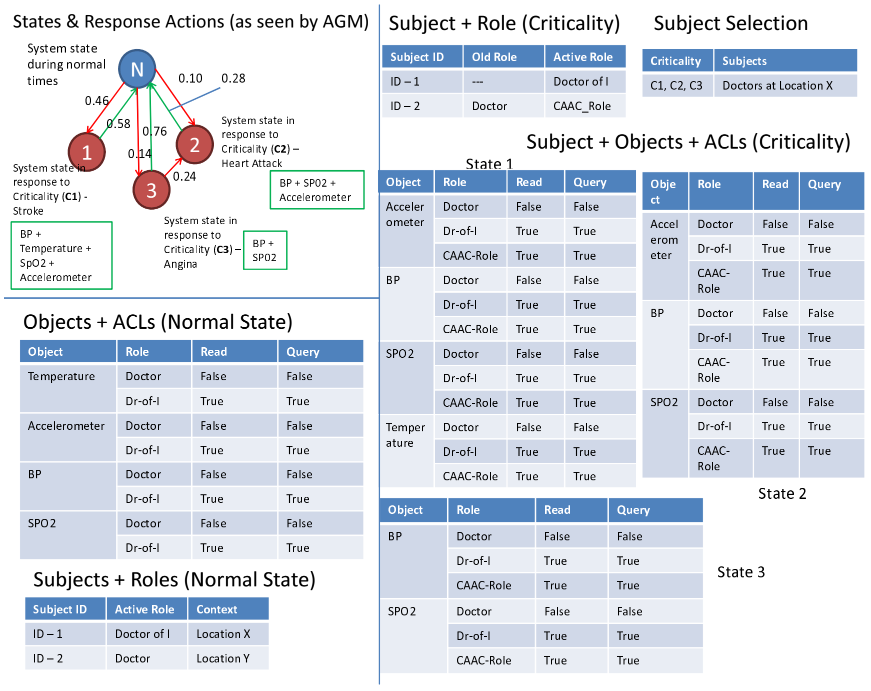


Fig. 38. CAAC Implementation Architecture

hypertensive patient I - ambient temperature, accelerometer, blood pressure and blood oxygen monitor. There are two subjects in this system ($ID-1$, $ID-2$), out of which subject $ID-2$ is the doctor of patient I. That subject therefore possesses the appropriate role which provides him with the ability to read and query all the objects under normal circumstances. The system considers three criticality situations which moves the PHMS from its normal state - 1) patient I has a stroke (in Location X) and her doctor is not nearby, 2) patient I has a heart attack and her doctor is not nearby, and 3) patient I angina pains and her doctor is not nearby. For each of these three states the CRET tool was executed and the response actions were determined. The response actions are then specified using CDL for CAAC to facilitate before the start of the operation of Ayushman. We have listed the response actions associated with each RL in the state model. The top left of the diagram shows the system states, CLs and RLs at each state of the system. The probabilities of the CLs and RLs have been obtained from various studies on occurrences of emergencies and response errors for medical ailments [19] [59]. In this simple scenario, the chosen RL chosen at each state that takes it directly to the normal state. Given our simple state model, the Q-value based metric for identifying the RL at any given state degenerates to the case of choosing the only RL available at each state. During the operation of


States & Response Actions (as seen by AGM)
System state during normal times
N
1
2
3
0.46
0.58
0.76
0.14
0.10
0.28
0.24
System state in response to Criticality (C1) - Stroke
BP + Temperature + SpO2 + Accelerometer
System state in response to Criticality (C2) – Heart Attack
BP + SP02 + Accelerometer
System state in response to Criticality (C3) – Angina
BP + SP02

Objects + ACLs (Normal State)

Object	Role	Read	Query
Temperature	Doctor	False	False
	Dr-of-I	True	True
Accelerometer	Doctor	False	False
	Dr-of-I	True	True
BP	Doctor	False	False
	Dr-of-I	True	True
SPO2	Doctor	False	False
	Dr-of-I	True	True

Subjects + Roles (Normal State)

Subject ID	Active Role	Context
ID – 1	Doctor of I	Location X
ID – 2	Doctor	Location Y

Subject + Role (Criticality)

Subject ID	Old Role	Active Role
ID – 1	---	Doctor of I
ID – 2	Doctor	CAAC_Role

Subject Selection

Criticality	Subjects
C1, C2, C3	Doctors at Location X

Subject + Objects + ACLs (Criticality)

State 1

Object	Role	Read	Query
Accelerometer	Doctor	False	False
	Dr-of-I	True	True
	CAAC-Role	True	True
BP	Doctor	False	False
	Dr-of-I	True	True
	CAAC-Role	True	True
SPO2	Doctor	False	False
	Dr-of-I	True	True
	CAAC-Role	True	True
Temperature	Doctor	False	False
	Dr-of-I	True	True
	CAAC-Role	True	True

Object	Role	Read	Query
Accelerometer	Doctor	False	False
	Dr-of-I	True	True
	CAAC-Role	True	True
BP	Doctor	False	False
	Dr-of-I	True	True
	CAAC-Role	True	True
SPO2	Doctor	False	False
	Dr-of-I	True	True
	CAAC-Role	True	True

State 2

Object	Role	Read	Query
BP	Doctor	False	False
	Dr-of-I	True	True
	CAAC-Role	True	True
SPO2	Doctor	False	False
	Dr-of-I	True	True
	CAAC-Role	True	True

State 3

Fig. 39. CAAC Operation Example

Ayushman, if a criticality is detected, the CAAC model identifies the response actions of the current (critical) state. Occurrence and detection of criticalities in our implementation were implemented as simple timed events.

The top-right corner of the diagram shows the rules for selecting subjects for each criticality scenario. In this case both are dynamic selection in nature and the doctor located at the patient's side (location X) is to be chosen. Therefore in this example, subject $ID - 2$ will be chosen in the event of any of the criticalities. Once the subject is chosen, the ACLs of the appropriate objects and the chosen subject is updated by CAAC (using the CDL specifications) with the required set of privileges, depending upon the state the system is in. As the system changes state, the ACLs of the objects change as well, and finally returning to the default form (as depicted in the bottom-left of the diagram) once the system is in normal state, and subject will go back to the role it had before criticality. The ACLs of the objects will also return to the default form in the event that criticalities cannot be controlled any-more, in which case no actions will be returned by CRET.

5.6. Lessons Learned

The implementation of PSKA and CAAC on Ayushman introduced many challenges. Below we list some of the lessons learned in overcoming these challenges.

- *Platform Selection*: The platform selection has to be done with an eye of future capabilities to be introduced. Our initial choice of the hardware was focused toward ensuring that efficient health data collection and interfacing with medical devices. However, as security features were added, the computation and storage requirements increased dramatically. Further, the motes used lacked the floating point arithmetic needed for computing FFTs. With motes being increasingly used for signal processing applications, we believe that the underlying system should be able to natively support them, so that application developers can focus on system design issues.

- *AGM Probability Generation*: An important requirement in implementing CAAC is to determine the probabilities for the CLs and MLs in AGM. System administrator have to consider the ailments that are being monitored and the general health of the patient and generate the appropriate probabilities. Further, the AGM the optimal Q-value based RL computation metric can face state-space explosion problem. Simpler heuristics such as choosing the RL with the highest probability (greedy approach) from any state can be used ameliorate the situation [81].

- *Platform Independence*: As technology keeps changing and new tools are available everyday, ensure that the infrastructure being built is as technology independent as possible. The use of XML based message exchange and for specifying the access control primitives is one such example which allows us to implement different components of Ayushman using different platforms and programming languages. Platform independence allows easier addition of features or improve the performance of the system.

- *Flexibility*: As much as possible automate every step of the implementation especially configurability and resetting process. This speeds up the debugging and fault management at the time of deployment. It also makes the overall system flexible to be deployed in diverse environments, and ideal property for a system with Ayushman's goals.

- *Infrastructural Issues*: As much as possible keep utilize devices with multi-model communication capability. We faced considerable problems in making the PDA communicate with the central server using the ASU Wifi infrastructure (over the Internet). This is because the PDA's MAC was not registered with the campus network which therefore prevented it from connecting with any machine within its domain. To overcome such problems we had to perform the base station to central server communication using an ad-hoc Wifi connection. With multi-modal devices we can switch to a different communication technology if one is not working.

5.7. Example Usage Scenario

We conclude this section by providing an example scenario which demonstrates how PSKA and CAAC can be used together to provide secure health monitoring. Consider a person who is being monitored by the Ayushman secure health monitoring system consisting of medical monitoring sensors such as oximeter and blood pressure monitors and ambient sensors such as accelerometers for detecting patient movement. The sensors are programmed to monitor the patient and send their data over to a base station implemented in a PDA carried by the patient. The data from the sensors is sent over a wireless channel to the base station.

The sensors first exchange a list of physiological signals that they can measure with the base station, and then they use it to agree on a common key using PSKA. The base station is assumed to be able to measure a subset of physiological signals itself. In order to perform the key agreement, the base station is interfaced with the patient's body and it can then execute PSKA and agree on a common key with each of the sensors on the patient's body. This interfacing of the base station with the patient's body is accomplished in a non-intrusive manner while the patient is holding the PDA. The successful use of PSKA also authenticates the sensors as they share a key with an entity that is on the same patient. Once the key is established, then the sensors can perform secure data communication with the base station. The use of PSKA makes the entire system plug-n-play, as the sensors now only have to be deployed to agree on a key and communicate securely using the key. No manual initialization needs to be done. The data sent by the sensors is collected by the base station, and forwarded through gateway entities toward the central server located at a medical facility. This forwarding may utilize the Internet infrastructure, and consequently can establish a secure tunnel between the base station and the central server.

At the central server, the patient data is added to the patient's Electronic Health Records (EHR) and can be further processed to derive specific information about the patient's health state. Caregivers who want to access the patient's data connect to the central server and submit query for the data or receive real-time inputs on the patient's health. The central server provides only that information about the patient which they are authorized to see. For example patient's detailed health information is available only to the caregiver assigned to the patient. If the patient being monitored suddenly becomes very sick, the central server processing this information will generate an alarm (the parameters for which are set in advance by health care provider) for the patient's caregiver who depending on the situation may dispatch an ambulance (from the care facility) to the patient's location. The central server will now switch to the CAAC mode, in which case, it determines the first responders dispatched to the scene (this information can be obtained from a central database, provided by the caregiver themselves or some other means) and provides them with access to the entire health information belonging to the patient. The first responders arriving at the scene can connect to the central server can query the patient's past data and current data while providing first aid. The patient's health information is available to the responders till the patient is brought to the care facility at which point the patient's caregiver takes over.

6. CONCLUSIONS AND FUTURE WORK

In this dissertation, we explored security solutions for the next generation of embedded systems called *Cyber-physical systems*. CPSs are networked, monitoring and actuation platforms, deeply embedded in specific physical processes. They introduce a level of automation in managing physical processes that has previously not been achieved. An important characteristic of CPSs is the level of detail of information they collect (about the physical process) in order to carry out their tasks. This work focused on developing solutions for securing this information collected by CPSs.

We present a novel security paradigm for CPSs called *Cyber-Physical Security* solutions, which takes into account the environmentally-coupled nature of CPSs in its operation. CYPSec solutions provide many interesting properties which traditional approaches, when applied to CPS domain, do not provide such as *usable security*, and *emergent properties*. In this dissertation, we proposed CYPSec solutions for two diverse domains:

- **Physiological Signal based Key Agreement :** PSKA dealt with providing usable (plug-n-play and transparent) security for communication within a BAN on a PHMS. It used simple, frequency domain features derived from common physiological signals such as plethysmogram and electrocardiogram, from the body of the host, in order to establish a secure channel (by distributing cryptographic keys) between sensors in the BAN. The distinctive and time-varying nature of the physiological features ensured that PSKA is secure from compromise based on physiological signals from another person or past physiological data from the host itself. PSKA was analyzed thoroughly based on prototype implementations using Matlab and actual sensor hardware on the *Ayushman* medical monitoring test-bed, and shown to be not only viable but also efficient in terms of sustainability using existing energy scavenging techniques.

- **Criticality Aware Access Control :** CAAC involved developing adaptive and proactive access control model for smart-infrastructure especially during criticalities. It used a Action Generation Model which considered the probabilities associated with managing the criticalities in order to determine the best set of response actions for a given set of criticalities. Each of these actions was then enabled without anyone subject explicitly requesting for it. The response actions are changed dynamically as the criticalities within the system change. We looked at a detailed case study for CAAC's deployment on a smart-oil rig platform and also demonstrated its viability by implementing a prototype on the Ayushman medical monitoring test-bed for managing health related criticalities.

In the rest of the chapter, we discuss other research problems related to our work.

6.1. Research Problems in PSKA

In this work, we presented a study of combining physiological signal based features and cryptographic primitives for securing key agreement in BANs. In this future, this work can be expanded in three possible directions - *minimizing overhead, enhance functionality and updated design*. The first, deals with making PSKA viable for mote and sub-mote architectures, while the second, involves improving its capabilities in terms of the types of keys distributed and the last one deals with keeping the basics of PSKA - combining signal processing and cryptographic primitives - while developing a completely new approach for key agreement.

6.1.1. Minimizing Overhead

PSKA protocol uses signal processing primitives such as FFT and peak detection along with mathematical routines such as polynomial generation and evaluation, to function. These can be expensive for platforms (memory-wise) such as Crossbow motes on which we have implemented PSKA. Indeed many older generations of Crossbow motes (Mica-dot, Mica, Mica2 motes) cannot handle the requirements of the protocol. Similarly, the use of fuzzy vault requires the generation and transmission of large number of chaff points, which adds considerable communication overhead for the key agreement process. One way of reducing the vault size is to use larger polynomial orders. A possible reason for this is the many approximations we have had to perform in implementing the signal processing primitives, for example use of binary fixed point representation rather than IEEE 754 for representing floating point numbers. That is, we have to make our signal processing and mathematical routines very efficient in order to be able to handle the extra computational requirements of such a change. Another possible approach is to develop a customized hardware board which can implement PSKA and thus reduces the overhead on the mote itself. The work done in [9] is the first step in this regard. What is ultimately required is the development of techniques which make PSKA efficient for mote and sub-mote architectures, as it is here PSKA will eventually be deployed.

6.1.2. Enhance Functionality

At this point PSKA only performs pair-wise key agreement between sensor in a BAN. This may be sufficient for smaller networks, but for managing very large networks, we need to incorporate group-wise and network-wise key agreement capability as well. An important line of enquiry could therefore be on extending PSKA to enable usable group and network key agreement with minimal increase in the communication, computation and memory overhead. The work on PSKA began with the goal of eliminating the need for communication during key agreement between sensors in a BAN by using synchronously measured physiological signal features as keys. However, the topographic specificity of the physiological signals made is very difficult to achieve this. One way to reduce the effects of topographic specificity was to use frequency domain features. Therefore, we believe it is worth investigating other signal processing and cryptographic techniques which can extend PSKA to perform key distribution without any form of communication exchange.

6.1.3. Updated Design

The last alternative is to re-design PSKA to make it more efficient and enhance its functionalities. Some of the possible avenues that can be investigated in this regard include - 1) Investigation of other (possibly simpler) physiological signal based features rather than FFT peak values and indexes for executing PSKA which might have greater correlation and need not use complex cryptographic primitives as fuzzy vault; 2) Identification of newer physiological signals which show lower topographic specificity; and 3) Investigation of the use of physiological process models (human body modeling) to determine the effectiveness of certain physiological signals to others in being used with PSKA.

6.2. Research Problems in CAAC

In this work, we presented a study of providing adaptive and proactive access control for criticality management in smart-infrastructures. In the future, this work can be expanded in two possible directions - *enhancing the action generation model* and *enhancing the access control*. The first one deals with improving the action generation model so that response actions can be generated online rather than pre-computed before the system is deployed, while the second deals with improving the access control model so that the response actions can be executed in a timely manner.

6.2.1. Enhancing the Action Generation Model

The AGM is lifeline of the CAAC model, as it determines the best response actions that need to be taken given the set of criticalities active within the system. However, the way the model is structured, it is difficult to execute it online during the operation of the system due to the inherently large number of states the system has to deal with, given the criticalities within the system. An off-line execution of the AGM may not be sufficient as in many cases where it is difficult to predict the states the system is in or the criticalities that the system may face. Therefore one open line of enquiry is to determine the response actions on-the-fly during the operation of the smart-infrastructure. Further, the success of CAAC depends largely upon the probabilities of the CLs and MLs in the AGM. However, it has been very difficult to obtain these values for a given system, as statistics are not collected and are difficult to come by. The availability of a common database with statistics for emergencies and errors in response would be very useful. One of the promising areas that need to be focused on is how to make such a database possible and available. Finally, the model also needs to take into account the constantly changing window-of-opportunity for all the criticalities within the system in computing the response actions.

6.2.2. Enhancing the Access Control

The access control aspect of CAAC can be improved considerably in order to improve its efficacy. At this time, the CAAC is very strict in restricting access during emergencies for actions which have no bearing on the emergency itself. This needs to be relaxed a little as during emergencies, it is not prudent to stop the functioning of the entire system. If such authorizations for non critical parts of the system are considered, then conflicts between privileges for a subject on different parts of the system, one under criticality and the other not, have to be considered and addressed. Additionally, in order to improve the dynamic nature and efficiency of CAAC, subjects in the system might be given different set of privileges for handling multiple criticalities simultaneously, which is not possible in the current model. In such cases, subjects cannot be given one privilege or role, but multiple privileges and roles need to be active for a subject simultaneously.

However, this might increase the complexity of authorization management considerably. A possible line of investigation could be to determine how to achieve this efficiently.

6.3. Future Work in Cyber-Physical Security Solutions for CPS

Apart from the aforementioned improvements to the solutions proposed, CYPSec solutions in general need to address a few important issues to make them viable. In this section we present some of the potential issues that need to be considered while deploying CYPSec solutions.

Notion of Time : The close coupling of CYPSec solutions with the environment also brings to fore an important characteristic which traditional security approaches do not seem to consider - the *notion of time*. Traditional computing usually ignores the notion of time by abstracting the physical process [64]. Security solutions developed for such computing systems also follow the same philosophy (i.e. their execution is treated as instantaneous for practical purposes limited only by the rate at which the input is generated). The execution of CYPSec solutions is not instantaneous as it is dependent upon a physical process. Physical processes do not execute in an instantaneous manner and usually take a finite, tangible amount of time. Consequently, CYPSec solutions cannot be assumed to execute instantaneously as it has to wait for specific inputs from the environment. Designers of software for CPS utilizing CYPSec solutions, have to consider this temporal semantics in their design. For example, while using PPG as the signal of choice, PSKA has to wait for about 13 seconds to collect the requisite number of samples. Further, the extensive processing requirements for key hiding and un-hiding further increases the latency to about 20 seconds [128].

Increased Complexity : CYPSec solutions usually require integrating primitives from multiple domains. This can potentially increase the complexity of the solution considerably. Given the presence of bottleneck (low capability) entities in a CPS, this can pose considerable issues in executing them a timely manner. Care therefore has to be taken that CYPSec solutions do not prevent them from performing their basic tasks. Therefore effort has to be put forth toward optimizing the implementation of any complex primitives required by CYPSec. However, given the dynamic nature of the physical process it might be possible that in some cases the precision of implementation can be reduced without significantly affecting behavior of the solution. A case in point is our implementation of PSKA on FPGAs and Crossbow Motes (http://www.xbow.com) for which we had to introduce many approximations (for floating point and complex arithmetic) in FFT and other signal processing computation which enabled us to emulate our Matlab-based benchmark results without sacrificing the security of the approach [10].

Non-determinism : One of the primary requirements of cyber-physical systems is that they need to be reliable and their operation predictable, that is, the behavior of the system is predictable under all scenarios [64]. CYPSec solutions too have to adhere to this requirement. This however is not easy to ensure for CYPSec solutions and dynamic nature of the physical process. A case in point is the key agreement using PSKA. The use of physiological signals based features introduces a level of non-determinism into the key agreement process which is difficult to overcome. It is possible with PSKA, that the physiological features between two sensors are not significantly similar to enable successful unlocking of the key. In such cases the whole process has to be repeated. CYPSec solutions therefore have to be aware of this phenomenon and compensate for it either by providing for repeating the process (as in PSKA), or deploying techniques which can handle such non-determinism. Care has to be taken that the compensatory mechanisms do not increase overall complexity of the solution.

Mixed Criticality: CPS can be mixed-critical systems with both critical (those that perform critical computations or those that interact with the physical process) and non-critical components [2]. Interaction between these two components has to be carefully considered in order to ensure the safe operation of the system. CYPSec solutions for mixed-critical CPSs which use both critical and non-critical components to function have to be aware of this difference and ensure that security provision does not affect the operation of the critical aspects of the system. One approach for handling the mixed critical nature is to use formal methods to formally verify the behavior of CYPSec solutions under different operational conditions of the system.

REFERENCES

[1] Crossbow technologies inc. http://www.xbow.com.

[2] Mixed Critical Systems. `http://www.nitrd.gov/about/blog/white\_papers/20-Mixed\_Criticality\_Systems.pdf`.

[3] Smith medical external oem oximetry solutions. http://www.smithsoem.com/applications/patientmon.htm.

[4] Suntech advantage oem bp module. http://www.suntechmed.com.

[5] *World Population Ageing: 1950-2050*. United Nations, Department of Economic and Social Affairs, Population Division.

[6] F. Adelstein, S. K. S. Gupta, G. Richard, and L. Schwiebert. *Fundamentals of Mobile and Pervasive Computing*. McGraw Hill, 2005.

[7] Aging-Report2006. The impact of the aging population on the health workforce in the united states: Summary of key findings. Center for Health Workforce Studies School of Public Health, University at Albany, March, 2006.

[8] C. A. Ardagna, S. D. C. di Vimercati, T. Grandison, S. Jajodia, and P. Samarati. Regulating Exceptions in Healthcare using Policy Spaces. July 2008. In Proc. of the 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security.

[9] A. Banerjee, K. Venkatasubramanian, and S. K. S. Gupta. Challenges of implementing cyber-physical security solutions in body area networks. April 2009. In Proc. of 4th International Conference on Body Area Networks.

[10] A. Banerjee, K. Venkatasubramanian, and S. K. S. Gupta. Challenges of Implementing Cyber-Physical Security Solutions in Body Area Networks. April 2009. In Proc. of International Conference on Body Area Networks (BodyNets 2009).

[11] J. E. Bardram, R. E. Kjaer, and M. O. Pedersen. Context-Aware User Authentication - Supporting Proximity-Based Login in Pervasive Computing. pages 107–123, 2003. In Proc. 5th International Conference on Ubiquitous Computing.

[12] M. J. Bass and C. M. Christensen. The future of the microprocessor business. *IEEE Sepctrum*, 39(4):34–39, April 2002.

[13] C. Bettini, S. Jajodia, X. Wang, and D. Wijesekera. Obligation Monitoring in Policy Management. 2002. In Proc. of the 3rd International Workshop on Policies for Distributed Systems and Networks.

[14] A. Bhargav-Spantzel, A. Squicciarini, and E. Bertino. Privacy Preserving Multi-factor Authentication with Biometrics. October 2006. In Proc. of the Second ACM Workshop on Digital Identity Management.

[15] M. Bishop. *Computer Security: Art and Science*. Addison-Wesley Professional, 1st edition, 2002.

[16] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. pages 275–286, May 1994. In Proceedings of Eurocrypt.

[17] A. A. Cárdenas, S. Amin, and S. Sastry. Research challenges for the security of control systems. pages 1–6, 2008. In Proc. of the 3rd conference on Hot Topics in Security (HOTSEC'08).

[18] D. Carman, B. Matt, and G. Cirincione. Energy-efficient and low-latency key management for sensor networks. December 2002. In Proceedings of 23rdth Army Science Conference.

[19] H. Chan, R. Anderson, and A. Perrig. Key infection:smart trust for smart dust. May 2004. In Proceedings of IEEE International Conference on Network Protocols.

[20] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor network. pages 197–213, May 2003. In Proceedings of IEEE Symposium of Research in Security and Privacy.

[21] P. S. Chan, H. M. Krumholz, G. Nichol, and B. K. Nallamothu. Delayed Time to Defibrillation after In-Hospital Cardiac Arrest. *The New England Journal of Medicine*, 358(1):9–17, January 2008.

[22] E.-C. Chang, R. Shen, and F. W. Teo. Finding the original point set hidden among chaff. In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 182–188, 2006.

[23] S. Cherukuri, K. Venkatasubramanian, and S. K. S. Gupta. BioSec: a biometric based approach for securing communication in wireless networks of biosensors implanted in the human body. pages 432–439, October 2003. In Proceedings of Wireless Security and Privacy Workshop 2003.

[24] S. Choi, S.-J. Song, K. Sohn, H. Kim, J. Kim, J. Yoo, and H.-J. Yoo. A low-power star-topology body area network controller for periodic data monitoring around and inside the human body. pages 139–140, 2006. ISWC.

[25] R. Clark. e-consent: A critical element of trust in e-business. 2002. In Proc. 15th Bled Electronic Commerce Conference.

[26] A. Corradi, R. Montanari, and D. Tibaldi. Context-based Access Control Management in Ubiquitous Environments. pages 253–260, September 2004. In Proc.of 3rd IEEE International Symposium on Network Computing and Applications.

[27] M. J. Covington, W. Long, and S. Srinivasan. Secure Context-Aware Applications Using Environmental Roles. May 2001. In Proc. of 6th ACM Symp. on Access Control Models and Technology.

[28] D. Halperin and T.S. Heydt-Benjamin and B. Ransford and S. S. Clark and B. Defend and W. Morgan and K. Fu and T. Kohno and W.H. Maisel. Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses. 2008. In Proc. of IEEE Symposium on Security and Privacy.

[29] J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer Verlag, 2002.

[30] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The Ponder Specification Language. January 2001. In Proc. Workshop on Policies for Distributed Systems and Networks.

[31] W. Diffie and M. E. Hellman. New directions in cryptography. ieee transactions on information theory. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[32] E. Dijkstra. Guarded Commands, Non Determinacy and Formal Derivation of Programs. *Communications of the ACM*, 18(8), 1975.

[33] G. D. DiMattia, I. K. Faisal, and P. R. Amyotte. Determination of human error probabilities for offshore platform musters. *Journal of Loss Prevention in the Process Industries*, 18:488–501, 2005.

[34] A. DroitCour. Non-contact measurement of heart and respiration rates with a single-chip microwave doppler radar, 2006. PhD Thesis.

[35] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili. A pairwise key predistribution scheme for wireless sensor networks. *ACM Transaction on Information Systems Security*, 8(2):228–258, 2005.

[36] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He. Demo-software-based sensor node energy estimation. pages 409–410, Nov 2007. In Proc. 5th ACM Conference on Embedded Networked Sensor Systems.

[37] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 2(34):469–472, 1985.

[38] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. volume 36, pages 147–163, 2002. In Proceedings of the 5th symposium on Operating systems design and implementation.

[39] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. pages 41–47, November 2002. In Proceedings of the 9th ACM conference on Computer and Communications Security.

[40] G. F.Adelstein, S.K.S.Gupta and L.Schwiebert. *Fundamentals of Mobile and Pervasive Computing*. McGraw Hill, 2005.

[41] A. Ferreira, R. Cruz-Correia, L. Antunes, P. Farinha, E. Oliveira-Palhares, D. W. Chadwick, and A. Costa-Pereira. How to Break Access Control in a Controlled Manner. 2006. In Proc. of 19th IEEE Symp. on Computer-Based Medical Systems.

[42] C. K. Georgiadis, I. Mavridis, G. Pangalos, and R. K. Thomas. Flexible Team-Based Organizational Access Control using Contexts. May 2001. In Proc. of 6th ACM Symp. on Access Control Models and Tech.

[43] E. F. Greneker. Radar sensing of heartbeat and respiration at a distance with applications of the technology. *IEE Radar*, 449:150–154, 1997.

[44] S. K. S. Gupta, T. Mukherjee, and K. Venkatasubramanian. Criticality Aware Access Control Model for Pervasive Applications. March 2006. In Proc. of the 4th IEEE Conference on Pervasive Computing.

[45] D. Halperin and et. al. Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses. 2008. In Proc. of IEEE Symposium on Security and Privacy.

[46] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen. The gator tech smart house: a programmable pervasive space. *Computer*, 38(3):50–60, March 2005.

[47] K. Hendrix, S. Mayhan, D. Lackland, and B. Egan. Prevalence, Treatment, and Control of Chest Pain Syndromes and Associated Risk Factors in Hypertensive Patients. *American Journal of Hypertension*, 18(8):1026–1032, May 2004.

[48] HIPAA-Report2003. Summary of HIPAA Health Insurance Probability and Accountability Act. US Department of Health and Human Service, May 2003.

[49] J. Hu and A. C. Weaver. A Dynamic, Context-Aware Security Infrastructure for Distributed Healthcare Applications. August 2003. In Proc. of Conference Pervasive Security, Privacy and Trust.

[50] Q. Huang, J. Cukier, H. Kobayashi, B. Liu, and J. Zhang. Fast authenticated key establishment protocols for self-organizing sensor networks. pages 141–150, 2003. In Proceedings of the 2nd ACM international conference on Wireless Sensor Networks and Applications.

[51] N. institute˜of˜standards˜and technology. Fips 180-2, secure hash standard, federal information processing standard (fips), publication 180-2. Technical report, DEPARTMENT OF COMMERCE, August 2002.

[52] J. B. D. Joshi, E. Bertino, and A. Ghafoor. Analysis of Expressiveness and Design Issues for a Temporal Role Based Access Control Model. *Transactions on Dependable and Secure Computing*, April-June 2005.

[53] E. Jovanov, A. Milenkovic, C. Otto, and P. C. de Groen. A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation. *SJournal of NeuroEngineering and Rehabilitation*, 2(6), March 2005.

[54] A. Juels and M. Sudan. A Fuzzy Vault Scheme. page 408, 2002. In Proc. of IEEE Intl. Symp. on Inf. Theory.

[55] L. Kagal, T. Finin, and A. Joshi. A Policy Language for A Pervasive Computing Environment. June 2003. In Proc. of IEEE 4th International Workshop on Policies for Distributed Systems and Networks.

[56] C. Kaufman, R. Perlman, and M. Speciner. *Network Security: Private Communication in a Public World*. Prentice Hall, 2nd edition, 1997.

[57] N. Kern and B. Schiele. Multi-sensor activity context detection for wearable computing. Eindhoven, The Netherlands, November 2003. In Proceedings of European Symposium on Ambient Intelligence.

[58] A. Kholmatov and B. Yanikoglu. Realization of correlation attack against the fuzzy vault scheme. volume 6819. SPIE, 2008.

[59] U. N. Khot, M. B. Khot, C. T. Bajzer, S. K. Sapp, E. M. Ohman, S. J. Brener, S. G. Ellis, A. M. Lincodd, and E. J. Topol. Prevalence of Conventional Risk Factors in Patients With Coronary Heart Disease. *The Journal of American medical Association*, 290(7), August 2003.

[60] C. Kuo, M. Luk, R. Negi, and A. Perrig. Message-in-a-bottle: User-friendly and secure key deployment for sensor nodes. In *Proceedings of the ACM Conference on Embedded Networked Sensor System (SenSys 2007)*, October 2007.

[61] K. V. Laerhoven and H. W. Gellersen. Spine versus porcupine: A study in distributed wearable activity recognition. pages 142–149, Oct 2004. In Proceeding of 8th International Symposium on Wearable Computers.

[62] K. V. Laerhoven, N. Villar, and H. W. Gellersen. A layered approach to wearable textile networks. pages 62–67, Oct 2003. In Proceedings of the IEE Eurowearable.

[63] B. Lai, S. Kim, and I. Verbauwhede. Scalable session key construction protocol for wireless sensor networks. 2002. In Proceedings of IEEE Workshop on Large Scale RealTime and Embedded Systems.

[64] E. A. Lee. Cyber physical systems: Design challenges. In *International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*, May 2008. Invited Paper.

[65] E. A. Lee. Computing needs time. *Communications of the ACM*, 52(5):70–79, May 2009.

[66] Y. Liao and J. Moody. Constructing Heterogeneous Committees Using Input Feature Grouping. *Advances in Neural Information Processing Systems (S.A. Solla, T.K. Leen and K.-R. Muller (eds.))*, 2000.

[67] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. pages 52–61, October 2003. In Proceedings of the 10th ACM conference on Computer and communications security.

[68] D. Liu and P. Ning. Location-based pairwise key establishment for static sensor networks. pages 72 – 82, October 2003. In Proceedings of the 1st ACM Workshop on Ad-hoc and Sensor Networks.

[69] H. Liu, H. Motoda, and L. Yu. A Selective Sampling Approach to Active Feature Selection. *Artificial Intelligence*, 159(1–2):49 – 74, November 2004.

[70] J. Lobo, R. Bhatia, and S. Naqvi. A Policy Description Language. July 1999. In Proc. of Sixteenth National Conference on Artificial Intelligence.

[71] P. Lukowicz, U. Anliker, J. Ward, G. Trster, E. Hirt, and C. Neufelt. Amon: A wearable medical computer for high risk patients. pages 133–134, October 2002. In Proceedings of IEEE 6th International Symposium on Wearable Computers.

[72] E. C. Lupu and M. Sloman. Conflicts in Policy-Based Distributed Systems Management. *IEEE Transactions on Software Engineering*, 25:852–869, November 1999.

[73] D. Ma and G. Tsudik. A new approach to secure logging. *Trans. Storage*, 5(1):1–21, 2009.

[74] D. J. Malan, M. Welsh, and M. D. Smith. A public-key infrastructure for key distribution in TinyOS based on Elliptic Curve Cryptography. pages 71–80, October 2004. In Proceedings of IEEE 2nd International Conference on Sensor and Ad Hoc Communications and Networks.

[75] R. Mayrhofer. The candidate key protocol for generating secret shared keys from similar sensor data streams. *Springer Lecture Notes in Computer Science*, 4572:1–15, 2007.

[76] R. Mayrhofer and H. Gellersen. Shake well before use: Authentication based on accelerometer data. volume 4880, pages 144–161, 2007.

[77] C. McWilliams. The biophysical properties of the transdermal measurement, 2003. International Society of Electrodermologists (http://www.electrodermology.com/biophysprop.htm).

[78] S. Mehrotra, C. Butts, D. Kalashnikov, N. Venkatasubramanian, R. Rao, G. Chockalingam, R. Eguchi, B. Adams, and C. Huyck. RESCUE: Challenges in Responding to the Unexpected. pages 179–192, January 2004. In Proc. of SPIE.

[79] P. Mihailescu. The Fuzzy Vault For Fingerprints Is Vulnerable To Brute Force Attack. The Computing Research Repository (Informal Publication), `http://www.citebase.org/abstract?id=oai:arXiv.org:0708.2974`, Aug 2007.

[80] M. J. Moyer and M. Ahamad. Generalized Role Based Access Control. April 2001. In Proc. of 21st International Conference Distributed Computing System.

[81] T. Mukherjee and S. K. S. Gupta. A modeling framework for evaluating effectiveness of smart-infrastructure crises management systems. In *IEEE International Conference on Technologies for Homeland Security (HST)*, Waltham, MA, USA, 2008. IEEE Computer Society Press.

[82] T. Mukherjee and S. K. S. Gupta. CRET: a crisis response evaluation tool to improve crisis preparednesss. In *IEEE International Conference on Technologies for Homeland Security (HST) (To Appear)*. IEEE Computer Society Press, 2009.

[83] T. Mukherjee, K. Venkatasubramanian, and S. K. S. Gupta. Performance Modeling of Critical Event Management for Ubiquitous Computing Applications. October 2006. In Proc. 9th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems.

[84] K. Ouchi, T. Suzuki, and M. Doi. Lifeminder: A wearable healthcare support system using user's context. pages 791–792, July 2002. In Proceeding of 22th International Conference on Distributed Computing Systems Workshops.

[85] J. A. Paradiso and T. Starner. Energy scavenging for mobile and wireless electronics. *Pervasive Computing, IEEE*, 4(1):18–27, Jan.-March 2005.

[86] R. Paradiso, G. Loriga, and N. Taccini. A wearable health care system based on knitted integrated sensors. *IEEE Transactions of Information Technologies in Biomedicine*, 9(3):337 – 344, 2005.

[87] R. Paranjape, J. Mahovsky, L. Benedicenti, and Z. Koles'. The electroencephalogram as a biometric. *Electrical and Computer Engineering, 2001. Canadian Conference on*, 2:1363–1366, 2001.

[88] J. Park and R. Sandhu. Towards Usage Control Models: Beyond Traditional Access Control. pages 57–64, June 2002. In Proc. of 7th ACM Symp. on Access Control Models and Tech.

[89] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar. SPINS: security protocol for sensor networks. *Wireless Networks*, 8(5):521–534, September 2002.

[90] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar. SPINS: security protocol for sensor networks. *Wireless Networks*, 8(5):521–534, September 2002.

[91] R. D. Pietro, L. V. Mancini, and A. Mei. Random key assignment for secure wireless sensor networks. pages 62 – 71, October 2003. In Proceedings of the ACM Workshop on Security of Ad Hoc and Sensor Networks.

[92] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *Proc. of the 4th international symposium on Information processing in sensor networks*, 2005.

[93] C. C. Y. Poon, Y.-T. Zhang, and S.-D. Bao. A novel biometrics method to secure wireless body area sensor networks for telemedicine and m-health. *IEEE Communications Magazine*, 44(4):73–81, 2006.

[94] J. H. Pope, T. P. Aufderheide, R. Ruthazer, R. H. Woolard, J. A. Feldman, J. R. Beshansky, J. L. Griffith, and H. P. Selker. Missed Diagnoses of Acute Cardiac Ischemia in the Emergency Department. *The New England Journal of Medicine*, 342(16):1163–1170, April 2000.

[95] M. Poulos, M. Rangoussi, and E. Kafetzopoulos. Person identification via the eeg using computational geometry algorithms. pages 2125–2128, September 1998. In Proc. of the 9th European Signal Processing.

[96] D. Povey. Optimistic Security: A New Access Control Paradigm. pages 40–45, 2000. In Proc. of Workshop on New Security Paradigms.

[97] E. S. Reddy and I. R. Babu. Authentication using fuzzy vault based on iris textures. pages 361–368, 2008. In Proc. of the 2nd Asia Intl. Conf. on Modelling & Simulation.

[98] J. Reid, I. Cheong, M. Henricksen, and J. Smith. A novel use of rbac to protect privacy in distributed health care information systems. pages 403–415, 2003. In Proc. of 8th Australasian Conference on Information Security and Privacy.

[99] E. K. Reilly, E. Carleton, and P. K. Wright. Thin film piezoelectric energy scavenging systems for long term medical monitoring. In *BSN '06: Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks*, pages 38–41, 2006.

[100] E. Rissanen, S. Firozabadi, and M. Sergot. Towards a Mechanism for Discretionary Overriding of Access Control. April 2004. In Proc. of 12th International Workshop on Security Protocols.

[101] A. S. R. L. Rivest and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *IEEE Personal Communication*, 21(2):120 – 126, 1978.

[102] R. L. Rivest. The md5 message-digest algorithm (rfc 1321), 1992.

[103] R. L. Rivest. The RC5 encryption algorithm. pages 86 – 96, 1995. Workshop on Fast Software Encryption.

[104] M. Román, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. A middleware infrastructure for active spaces. *IEEE Pervasive Computing*, 1(4):74–83, 2002.

[105] G. Sampemane, P. Naldurg, and R. H. Campbell. Access control for Active Spaces. December 2002. In Proc. of 18th Annual Computer Security Applications Conference.

[106] R. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role Based Access Control Models. *IEEE Computer*, pages 38–47, February 1996.

[107] W. J. Scheirer and T. E. Boult. Cracking Fuzzy Vaults and Biometric Encryption. pages 1–6, Sept 2007. In Proc. of the Biometrics Symposium, 2007.

[108] B. Schenier. *Applied Crytpography*. John Wiley and Sons, 2nd Edition edition, 1996.

[109] B. Schneier. Description of a new variable-length key, 64-bit block cipher (blow- sh). 1993. In Proceedings of Fast Software Encryption.

[110] B. Schneier and J. Kelsey. Secure audit logs to support computer forensics. *ACM Trans. Inf. Syst. Secur.*, 2(2):159–176, 1999.

[111] L. Schwiebert, S. K. S. Gupta, and J. Weinmann. Research challenges in wireless networks of biomedical sensors. pages 151–165, July 2001. In Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking.

[112] C. Shankar, A. Ranganathan, and R. H. Campbell. An ECA-P Policy-based Framework for Managing Ubiquitous Computing Environments. July 2005. In Proc.of 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services.

[113] V. Shnayder, B. Chen, K. Lorincz, T. R. F. Fulford-Jones, and M. Welsh. Sensor Networks for Medical Care. April 2005. Harvard University Technical Report.

[114] S. Slijepcevic, M. Potkonjak, V. Tsiatsis, S. Zimbeck, and M. B. Srivastava. On communication security in wireless ad-hoc sensor networks. pages 139–144, June 2002. In Proceedings of IEEE 11th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises.

[115] M. Sloman and E. Lupu. Security and Management Policy Specification. *IEEE Network*, 16(2):10–19, March/April 2002.

[116] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. pages 172–194, 1999. In Proceedings of the 7th International Workshop on Security Protocols.

[117] V. Stanford. Pervasive health care applications face tough security challenges. *IEEE Pervasive Computing*, 8(12), 2002.

[118] J. A. Stankovic, I. Lee, A. Mok, and R. Rajkumar. Opportunities and obligations for physical computing systems. *IEEE Computer*, 38(11):23–31, November 2005.

[119] T. Starner and J. A. Paradiso. Human generated power for mobile electronics. In *Low Power Electronics Design*, pages 1–35. CRC Press, 2004.

[120] P. Tabuada. Cyber-physical systems: Position paper. October 2006. Workshop on Cyber-Physical Systems.

[121] Q. Tang. Thermal-aware scheduling in environmentally coupled cyber-physical distributed systems, 2008. PhD Thesis.

[122] U. Uludag, S. Pankanti, and A. K. Jain. Fuzzy Vault for Fingerprints. pages 310–319, July 2005. In Proc. of Audio- and Video-based Biometric Person Authentication.

[123] U. Varshney. Pervasive healthcare. *IEEE Computer*, 36(12):138–140, 2003.

[124] K. Venkatasubramanian, A. Banerjee, and S. K. S. Gupta. CAAC - an adaptive and proactive access control approach for emergencies for smart infrastructures. *ACM Transactions on Autonomous and Adaptive Systems Special Issue on Adaptive Security*.

[125] K. Venkatasubramanian, A. Banerjee, and S. K. S. Gupta. PSKA: Usable and secure key agreement scheme for body area networks. *IEEE Transaction on Information Technology in Biomedicine Special Issue on Wireless Health*.

[126] K. Venkatasubramanian, A. Banerjee, and S. K. S. Gupta. EKG-based key agreement in body sensor networks. April 2008. In Proc. of the 2nd Workshop on Mission Critical Networks.

[127] K. Venkatasubramanian, A. Banerjee, and S. K. S. Gupta. Plethysmogram-based secure inter-sensor communication in body area networks. November 2008. In Proc. of IEEE Military Communications Conference.

[128] K. Venkatasubramanian, A. Banerjee, and S. K. S. Gupta. Green and sustainable cyber physical security solutions for body area networks. June 2009. In Proc. of International Workshop on Body Sensor Networks (BSN 2009).

[129] K. Venkatasubramanian and et al. Ayushman: a wireless sensor network based health monitoring infrastructure and testbed. pages 406–407, June/July 2005. In Proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems.

[130] K. Venkatasubramanian and S. K. S. Gupta. Physiological value based efficient usable security solutions for body sensor networks. *ACM Transaciton on Sensor Network*.

[131] K. Venkatasubramanian and S. K. S. Gupta. Security for pervasive health monitoring sensor applications. pages 197–202, December 2006. In Proceedings of the 4th International Conference on Intelligent Sensing and Information Processing.

[132] K. Venkatasubramanian and S. K. S. Gupta. *Chapter 15: Security for Pervasive Healthcare*. In Security in Distributed, Grid, Mobile, and Pervasive Computing, ed. Yang Xiao. CRC Press, 2007.

[133] G. Virone, A. Wood, L. Selavo, Q. Cao, L. Fang, T. Doan, Z. He, and J. Stankovic. An assisted living oriented information system based on a residential wireless sensor network. April 2006. In Proceedings of the Transdisciplinary Conference on Distributed Diagnosis and Home Healthcare.

[134] H. Wang, Y. Zhang, and J. Cao. Ubiquitous Computing Environments and its Usage Access Control. May/June 2006. In Proc. of 7th Annual International Conference Digital Government Research.

[135] S. Warren and E. Jovanov. The need for rules of engagement applied to wireless body area networks. pages 979–983, 2006. In Proc. of 3rd IEEE Consumer Communications and Networking Conference.

[136] S. Warren, J. Lebak, and E. Jovanov. Lessons learned from applying interoperability and information exchange standards to a wearable point-of-care system. pages 101–104, 2006. In Proc. of 1st Transdisciplinary Conference on Distributed Diagnosis and Home Healthcare.

[137] A. C. Weaver. Biometric Authentication. *IEEE Computer*, 39(2):96–97, February 2006.

[138] B. J. West. *Where Medicine Went Wrong: Rediscovering the Path to Complexity*, volume 11 of *Studies of Non Linear Phenomena in Life Sciences*. World Scientific, 2006.

[139] M. White, B.Jennings, V. Osmani, and S. V. der Meer. Context Driven User Centric Access Control for Smart Spaces. 2005. In Proc. of IEEE International Workshop on Intelligent Environments.

[140] E. Yeatman, D. O'Hare, C. Dobson, and E. Bitziou. Approaches to self-powered biochemical sensors for in-vivo applications. In *Proc. of 3rd International Conference on Body Area Networks*, pages 1–2, 2008.

[141] E. M. Yeatman. Rotating and gyroscopic mems energy scavenging. *Wearable and Implantable Body Sensor Networks, International Workshop on*, pages 42–45, 2006.

[142] S. Zhu, S. Setia, and S. Jajodia. LEAP+: Efficient security mechanisms for large-scale distributed sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(4):500–528, November 2006.

[143] B. Ziaie and K. Najafi. An implantable microsystem for tonometric blood pressure measurement. *Biomedical Microdevices*, 3(8):285–292, December 2001.