Multi-Channel Security Protocols in Personal Networks

Xin Huang
Wolfson College
University of Oxford

A dissertation submitted for the degree of
Doctor of Philosophy

Hilary Term 2014

Abstract

Personal computing devices are becoming more and more popular. These devices are able to collaborate with each other using wireless communication technologies, and then support many applications. Some interesting examples of these are healthcare, context-aware computing, and sports training.

In any such applications, security is of vital importance. Firstly, sensitive personal data is always collected in these applications, thus confidentiality is usually required. Secondly, authenticity and integrity of data or instructions are always critical; incorrect data or instructions are not only useless, but also harmful in some cases.

This thesis analyses the security requirements of personal networks, and develops a number of multi-channel security protocols. With the help of out-of-band channels, especially no-spoofing and no-blocking out-of-band channels, these protocols can bootstrap security in personal networks. In particular, three kinds of security protocols have been studied: protocols that use human-controlled channels, protocols that use visible light communications, and protocols that use intra-body communications. Interesting trade-offs have been discovered among communication, computation and security, resulting from different channel implementations and protocols.

Acknowledgements

First of all, I would like to take this opportunity to express my sincere gratitude to my supervisor Professor Bill Roscoe for his guidance throughout my time at Oxford. He has made many contributions to this work, as well as helping me secure funding from Oxford Martin School for my D.Phil. studies.

In addition, I would like to thank Professor Tingting Zhang, Dr. Qinghua Wang, Dr. Bangdao Chen, Dr. Andrew Markham, Dr. Long Nguyen, and Dr. Ronald Kainda for their valuable comments, support, and background studies. Thanks also go to my friends Xiao Ma, Shangyuan Guo, Rong Fu, Xiong Gao, Lei Han, Zheng Yan for many fruitful conversations and cooperation.

This work is supported by studentships from Oxford Martin School. Special thanks to all the colleagues from the Institute for the Future of Computing in Oxford Martin School.

Finally, I would like to thank my family and friends for their love and friendship. Without their support, this dissertation could never have been written.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Information technology is taking over our lives including many personal domains. A good example of this is in healthcare. In any such personal domain, and especially healthcare, security is of vital importance. This thesis analyses the security needs of personal networks, and develops a number of solutions.

1.1 Personal Networks

Personal networks (*PNs*) are wireless networks of wearable or implanted nodes. Each node is a tiny computing device. These nodes are able to collaborate with each other using Bluetooth, Zigbee and other wireless communication technologies.

It is helpful to clarify the relationship of personal networks to some closely related concepts. In this thesis, we consider *personal networks as enhanced body sensor networks*: personal networks use both sensors (e.g. a heartbeat sensor) and actuators (e.g. q pacemaker), but body sensor networks only use sensors. In addition, there are overlaps among personal networks, wireless sensor networks, body area networks, ubiquitous computing, pervasive computing, and mobile computing. Thus, some concepts and results from these related areas will also be used.

Personal networks have attracted many communities. Firstly, more and more researchers are studying this network. For example, from 2003 to 2012, the number of new search results of Google Scholar using key word "body sensor network" increased from 2 in 2003 to

around 1000 in 2012 [1]. In addition, many organizations, for example Institute of Electrical and Electronics Engineers, ZigBee Alliance and Bluetooth Special Interest Group, have established standards for these networks. Besides, many related corporations, for example, Shimmer Research[2], Moog Crossbow[3] and Silicon Laboratories[4], have been founded. Thus, these networks are promising technologies, and their security therefore becomes meaningful, especially, as they are widely used in the real world.

Applications Personal networks are bridges between human users, the environment, and application programs, shown in Fig. 1.1. They can interact with the human body, the surrounding environment, and application programs. They have the potential to revolutionize many fields including medical applications, context-aware computing[5], affective computing[6], sports training, and interactive gaming. We list several interesting examples below.

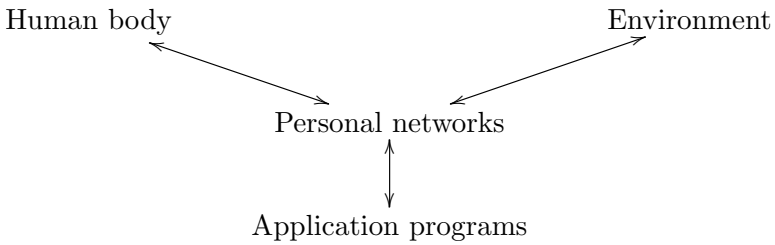


Figure 1.1: A simplified personal network application

Example (Emergency response system). Personal networks enable wireless monitoring and tracking of patients in emergency response scenarios. They use for example small vital sign sensors, motion sensors and location sensors, which are attached to patients. These sensors are connected via wireless communications; and they collect sensor data such as pulse rate, electrocardiography (ECG) signals, motion data, and location information. The sensor data is transmitted to doctors' mobile phones. Using the data, doctors can make quicker and more informed decisions. □

[1] Access time: 2013-05-15

[2] http://www.shimmer-research.com/, access time: 2013-07-06

[3] http://www.xbow.com/, access time: 2013-07-06

[4] http://www.silabs.com/pages/default.aspx, access time: 2013-07-06

[5] Context-aware computing refers to systems that can provide services based on context; one example of context is locations.

[6] Affective computing refers to systems that can provide services based on human affects.

Example (Cochlear implant system). Personal networks can be used in cochlear implant systems [105]. An external node firstly extracts features of sound. Secondly, it sends the features to an internal node in the ear over wireless channels. Thirdly, the internal node converts the features into electric currents for appropriate electrodes at the auditory nerve. Meanwhile, the internal node monitors critical electrical and neural activities, and transmits these activities back to the external node as feedback. These systems can help in the restoration of hearing. □

Example (Retinal implant system). Personal networks can be used in retinal implant systems [105]. An external video-capture node sends image data to the implanted node via a wireless channel. The implanted node translates the image into biphasic current pulses. Researchers at the Boston Retinal Implant Project have implanted a device in the eye of a pig; the system can potentially restore sight to approximately the 1.7 million people who suffer from retina-related diseases. □

Example (Affective computing). Personal networks enable dancers to express their feelings and moods by dynamically and automatically adjusting music and lighting in a dance environment. The music and lightning reflect the dancers' mood, while presenting their gestures and body movements [39]. These functionalities are realized using three layers. Layer 1 defines a sensor system that analyses and synthesizes physiological and pressure sensor signals. Layer 2 employs an intelligent system that controls light direction, colour and projected imagery, as well as music effects in order to portray a dancer's mood. Layer 3 translates the high-level adjustments made by the intelligent systems to an appropriate lighting control board, image rendering, and audio box commands. □

1.2 Motivation

Security mechanisms, especially the key establishment protocols, are important; however, there are still many problems in current mechanisms. The motivation is elaborated from two aspects: (1) the importance of security mechanisms, and (2) current mechanisms and problems.

1.2.1 Importance of Security Mechanisms

Security mechanisms are important. Effective key establishment protocols especially should be used. The reasons are explained below.

Importance of sensor data protection Personal networks bring new possibilities to many fields. One main reason is that these networks can provide detailed (sensor) data of users. For example, medical treatment can be more accurate, which is depending on the data. Thus, (sensor) data protection is the core task in personal networks.

Confidentiality. The confidentiality of the sensor data should be protected. Confidentiality of the data used in medical applications, for example sensor data in the emergency response system mentioned previously, is mandated by privacy laws and regulations (the Health Information and Portability Accountability Act and the European Union Directive 2002/58/EC). In addition, many other personal networks collect sensitive personal information. For example, the retinal implant systems can collect users' passwords, the confidentiality of sensor data in these networks is also very important.

Authenticity and Integrity. Sensor data authenticity and integrity are always critical. For example, compromised medical data in the emergency response system may lead to inappropriate treatment; wrong sensor data in the implant system makes the system useless; and when the sensor data in the affective computing system is maliciously modified, the whole performance would fail. Thus, compromised data may not only be useless, but may also be harmful.

Availability and Freshness. First of all, the data and the network should be available at any time. It means that attacks that exhaust service provider resources and the network bandwidth should be prevented. Meanwhile, the data, for example the heartbeat rate data, should be the latest.

Personal networks are vulnerable Personal networks have much vulnerability, which is described as follows. Firstly, since wireless radio communications are used, it is easy for an adversary A to eavesdrop, modify and block the messages. Secondly, the computation capability, memory and energy of wearable and implanted nodes are always limited. The

limitations mean that many effective but complex security techniques are not feasible, thus A has an advantage.

Importance of key establishment protocols Key establishment protocols are the basis of most security mechanisms. Secure communications essentially have two phases: trust establishment and secure data communication [9]. Trust establishment assures the legitimacy of communicating entities. It is usually based on cryptographic keys. Once keys are established, various cryptographic algorithms can easily help data to be transmitted securely. Therefore, key establishment protocols are the foundations of security mechanisms.

1.2.2 Current Mechanisms and Problems

Most current security mechanisms have some disadvantages when they are used in personal networks. They are stated briefly below (more discussions can be found in Section 2.6).

Mechanisms based on key pre-distribution schemes Mechanisms based on key pre-distribution schemes are the dominant methods used in wireless sensor networks. The procedure is generally as follows: firstly, secrets are deployed in a pre-deployment phase; secondly, symmetric keys are securely distributed using the secrets; thirdly, other security protections are provided using the symmetric keys.

However, these mechanisms have some disadvantages. For example, it is always difficult to update pre-deployed secrets even when they are compromised. In addition, adding new nodes or removing nodes is always troublesome. Besides, when nodes belong to different users, for example patients and doctors, they have to make the pre-deployed secrets public, which is not preferred by many users.

Mechanisms based on asymmetric keys Mechanisms based on asymmetric keys use a pair of public and private keys to initiate secure connections between nodes. However, most current studies focus on computational efficiency, and key distribution is widely ignored. The few solutions that focus on public key distribution also show some disadvantages. Some solutions depend on external trusted parties that are not always available. Others depend on location information that is only suitable for static networks.

Physical layer security mechanisms Physical layer security mechanisms mainly use the characteristics of wireless radio waves. However, they usually need multiple antennas. In addition, the key generation rates are always limited. Besides, group key establishments are not always possible.

Mechanisms using physiological values Mechanisms using physiological values to create random strings for security are designed specifically for personal networks. They use, for example, ECG characteristics and error correcting codes in order to generate symmetric keys in different nodes.

However, there are some problems. In using these mechanisms, each node is required to have, for example, a suitable ECG sensor, which is inconvenient and expensive. In addition, it is possible for an attacker A to get the physiological values from a non-contact camera; and then, A can generate the symmetric keys and compromise the whole system.

Remark. These mechanisms mainly focus on authenticity, integrity, confidentiality, and data freshness. However, availability is not carefully considered. One reason is that attacks on availability mainly focus on links between personal networks and the remote application servers, corresponding countermeasures are beyond the scope of personal networks in most cases. Additionally, attacks against physical links or nodes generally cannot be easily solved by using cryptography-based mechanisms, although they can be helpful (we will show that these mechanisms can be used to detect or alleviate such attacks).

1.3 Solution

We mainly aim to provide security mechanisms, especially key establishment protocols, in personal networks. These mechanisms should meet the following requirements:

- Authenticity. After running protocols, users can make sure that each node is authentic. Also, the generated security credentials are authentic.
- Integrity. The generated security credentials are not maliciously changed by attackers.
- Confidentiality. Session keys generated using these security credentials are only known by legal parties.

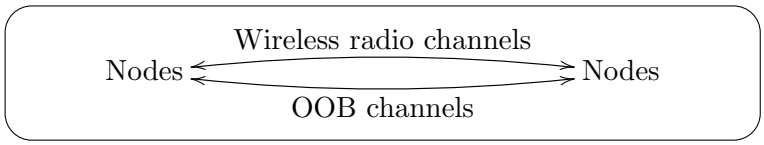


Figure 1.2: Channels in one network

- Freshness. Freshness of session keys should be guaranteed. Sensor data freshness is not considered here.
- Availability. Availability is not the main goal of a key establishment protocol, however, protocol should be designed that attacks against availability can be alleviated.

Below, we give the potential solution. In addition, we will list the challenges. Also, the scope of this thesis will be discussed.

Main solution Our main solution is a family of protocols based on multiple channels. The simplified scenario is shown in Fig. 1.2. In one personal network, in addition to wireless radio channels, many other channels, named out-of-band channels, are also available. Using proper out-of-band channels, message authenticity, integrity and freshness can be guaranteed; and then security protocols can be established.

Hypothesis Multi-channel security protocols will have higher utility in providing security in personal networks. The potential advantages of these mechanisms are

- the identities and security credentials can be easily updated after deployment (better than current key pre-distribution schemes)
- not dependent on a trusted third party or location information (better than current mechanisms based on asymmetric keys)
- they are not vulnerable to eavesdropper, for example, the eavesdropper uses a camera to record the authentication process and analyzes afterwards (better than current physiological values)
- they do not rely on devices that are not widely available in current personal networks; and protocol execution time is reasonably short (better than current physical layer

security mechanisms)

- they are efficient: user burden should be minimized.

Key aspects of multi-channel security protocols Out-of-band channels. Establishment of such channels are extremely important to these security protocols. These channels should hold certain security properties, for example, authenticity and availability. Also, their networking properties, for example channel speed, are also one important factor that should be considered in the protocol designing procedure.

User. Users are significant to these protocols. Without users, the trust link among devices can not be easily established. Generally, we assume users can prevent interference or attacks to out-of-band channels. Certainly, we will carefully choose out-of-band channels that make users feel easy to achieve such tasks.

Personal networks. Personal networks are the application or context of these protocols. Many assumptions to out-of-band channels are true only when they are established in personal networks. For example, it is reasonable to assume that users can notice that attackers is trying to block visible light, and then stop such attacks. This is because users can put two devices close enough, and make sure that these devices are within users' field of view; thus, users can be pretty sure that any interference can be easily noticed and stopped. However, if the scenario is not a personal network, such assumption is not always true.

Challenges Since most nodes only have constrained interfaces, which are typically LEDs and one single button, it is a challenge to find proper out-of-band channels. Additionally, it is interesting to design the corresponding security protocols that fulfil our aims mentioned previously.

Scope We focus on the security among the nodes in one personal network, since the security between a coordinator node of the network and a remote service centre (for example, a hospital) is a much easier problem. In addition, it has been demonstrated that asymmetric and symmetric cryptography are usable in sensor networks [70], which means that we concentrate on other aspects of the problem. Also, the usability is not included.

1.4 Contributions

In this section, we list the main contributions below.

Human interactive security protocols Human interactive security protocols (HISPs) are designed for bootstrapping security in wearable personal networks using human-controlled channels (HCCs). The details are as follows.

- Human interface devices and human-node interactions investigation.
- Analysis of HCC security and networking properties.
- HISPs and example protocols.
- ECDH[7] version of HISPs, which is a clan of HISPs that can eliminate man-in-the-middle attacks against ECDH.
- Identity binding protocols, a family of HISPs that can eliminate Sybil attacks.

Visible light communication channels and security protocols HVSPs[8] are protocols designed for bootstrapping security in wearable personal networks using HVLCs[9]. The details are as follows.

- VLCs[10] design and implementation.
- Analysis of HVLC security and networking properties.
- HVSPs and example protocols.
- HVSP-K[11], a clan of key establishment protocols that can reduce the burden of human users and the computation burden in slave nodes using HVLCs.
- Secure positioning system, which can eliminate location spoofing attacks found in [106] against the Skyhook positioning system.

[7] ECDH is an acronym for Elliptic Curve Diffie-Hellman
[8] H for human-controlled, V for visible light communication, SP for security protocol
[9] H for human-controlled, VLC for visible light communications
[10] VLC is an acronym for visible light communication.
[11] K for key establishment protocols

Intra-body communication channels and security protocols HBSPs[12] are protocols designed for bootstrapping security in wearable and implanted personal networks using HBCs[13]. The details are as follows.

- IBCs[14] design and implementation.
- Analysis of HBC security and networking properties.
- HBSPs and example protocols.
- SecPN, a demo personal network that provides a clan of key establishment protocols using HBCs.
- Multi-channel identity-based protocols, which can eliminate man-in-the-middle attacks and impersonation attacks against an identity-based protocol proposed by Hölbl and Welzer.

Review of personal networks and their security Personal networks, their security threats, and protection techniques are reviewed. The details are as follows.

- Enabling technologies including platforms and communication technologies.
- Applications proposed in recent years.
- Security threats and current security techniques, especially key establishment techniques.
- Standards including IEEE 802.15.4, Zigbee, Bluetooth, and 6LoWPAN.

1.5 Publications

This thesis is partly based on the following publications. The contributions of the author are also listed.

[12] H for human, B for body, SP for security protocol
[13] H for human, B for body, and C for communication channel
[14] IBC is an acronym for intra-body communication.

1. Xin Huang, Bangdao Chen, Andrew Markham, Qinghua Wang, Yan Zheng, and A.W. Roscoe, Human interactive secure key and ID exchange protocols in body sensor networks, IET Information Security, Special Issue on Trust and Identity Management in Mobile and Internet Computing and Communications, 2013.

   Contributions of the author: (1) protocol design; (2) performance evaluation.

2. Xin Huang, Xiao Ma, Bangdao Chen, Andrew Markham, Qinghua Wang, and A.W. Roscoe, Human Interactive Secure ID Management in Body Sensor Network, Journal of Networks. Vol. 7. No. 9. 2012.

   Contributions of the author: (1) protocol design; (2) performance evaluation.

3. Xin Huang, Shangyuan Guo, Bangdao Chen, and A.W. Roscoe, Bootstrapping Body Sensor Networks Using Human Controlled LED-Camera Channels, The 7th International Conference for Internet Technology and Secured Transactions (ICITST-2012), December 10-12 2012, London, UK.

   Contributions of the author: (1) design of human-controlled LED-camera channel; (2) protocol design; (3) performance evaluation.

4. Xin Huang, Rong Fu, Bangdao Chen, Tingting Zhang and A.W. Roscoe, User Interactive Internet of Things Privacy Preserved Access Control, The 7th International Conference for Internet Technology and Secured Transactions (ICITST-2012), December 10-12 2012, London, UK.

   Contributions of the author: (1) protocol design; (2) Internet of Things platform design; (3) context-aware K-anonymity policy design; (4) participated in privacy protection mechanisms investigation.

5. Xin Huang, Bangdao Chen and A.W. Roscoe, Multi-Channel Key Distribution Protocols Using Visible Light Communications in Body Sensor Networks, The Oxford University Department of Computer Science Student Conference 2012, 16th November, 2012, Oxford, UK.

Contributions of the author: (1) design of visible light communication channels; (2) protocol design.

6. Xin Huang, Qinghua Wang, Bangdao Chen, Andrew Markham, Riku Jäntti, and A.W. Roscoe, Body Sensor Network Key Distribution Using Human Interactive Channels, The 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies, October 26-29, 2011, Barcelona, Catalonia, Spain.

   Contributions of the author: (1) protocol design; (2) performance evaluation; (3) transportation of TinyECC library.

7. Qinghua Wang, Ilangko Balasingham, Miaomiao Zhang, and Xin Huang, Improving RSS-based ranging in LOS-NLOS scenario using GMMs, IEEE Communications Letters, vol. 15, no. 10, October 2011, pp. 1065-1067.

   Contributions of the author: (1) participated in the design of range estimation model based on radio signal strength in line-of-sight and non-line-of-sight scenario; (2) discussion of the usage of Gaussian mixture model.

1.6 Thesis Outline

The rest of this thesis is organized as follows. Chapter 2 reviews personal networks and their security. Chapter 3 introduces human interactive security protocols; and two use cases are provided in Chapter 4. Chapter 5 describes visible light communication and security protocols; and two use cases are provided in Chapter 6. Chapter 7 presents intra-body communication and security protocols; and two use cases are provided in Chapter 8. In Chapter 9, these protocols are evaluated. Finally, Chapter 10 synthesises the main contributions; and future works are proposed.

Chapter 2

Personal Networks and Security

In this chapter, we will give a survey of personal networks and their security. We will firstly investigate their network model, enabling platforms, communication technologies, and applications proposed in recent years. Secondly, we will review their security threats and current security mechanisms, especially key establishment techniques. Finally, we will introduce related standards including IEEE 802.15.4, Zigbee, Bluetooth, and 6LoWPAN.

Contents

2.1 Network Model

For the time being, we can categorize any node as either a coordinator node or a slave node. These two nodes are explained as follows.

- Coordinator node. A coordinator is a device that collects, aggregates and submits sensor data to applications, for example a smart phone running an appropriate application. This device is also called a body control unit or a coordinator node.

- Slave node. The first type of slave node is a device that gathers and reports sensor data. This device is also called a body sensor unit. The second type is actuator nodes that act according to data received from the sensors or through interaction with the user.

A general node is represented as N; the coordinator is C; and the slave node is S. In other words,

$$\mathsf{N} = \begin{cases} \mathsf{C}, & \text{coordinator node;} \\ \mathsf{S}, & \text{slave node.} \end{cases}$$

In this thesis, we assume that the group size gs is around 10 for practical reasons (There may be larger group sizes in some applications). This group size limitation is reasonable, because users will feel unconformable if there are too many nodes associated with their body.

2.2 Hardware and Software Platforms

Current platforms can be roughly categorized as general-purpose and dedicated nodes. General-purpose nodes include mobile phones and tablets. They usually have higher processing capabilities, a mature operating system, a richer set of networking protocols, and off-the-shelf software. However, they are energy-hungry in most cases. Dedicated nodes include Berkeley motes, Tmote Sky/Telosb motes, and the Arduino family. They typically use commercial off-the-shelf chips, and are energy-efficient. In personal networks, slave nodes are dedicated nodes in most cases; coordinator nodes may be dedicated or general-purpose nodes. In this section, we will focus on hardware and software platforms for dedicated nodes, because general-purpose computing devices are already being used in daily life.

Table 2.1: Examples of nodes' parameters

Node	Microcontroller	Flash memory	Transceiver	Sensor
Tmote Sky	8 MHz	1 MB	CC2420	Light, humidity, temperature
Arduino Uno	16 MHz	32KB	Xbee etc.	Various
Arduino Mega 2560	16 MHz	256 KB	Xbee etc.	Various
Arduino LilyPad	8 MHz	16 KB	Xbee etc.	Various

2.2.1 Single Node Architecture

Typical node architecture A typical node consists of a microcontroller, a memory component, a radio transceiver, several sensors, several actuators, and an energy source. Many low-cost nodes are commercially available. Tmote Sky[1], Arduino Uno[2], Arduino Mega 2560[3] and Arduino LilyPad[4] are four examples. Their parameters are listed in Table 2.1. As we can see, their capabilities are limited, thus very complex security mechanisms are not suitable.

Microcontroller A microcontroller is a self-contained system on a single integrated circuit. This system contains processor cores, memory components, and programmable input/output peripherals. The main parameter of the system is the processing speed, which to a large extent determines the execution time of a program.

Most current microcontrollers are embedded in other systems. For example, current mobile phones have at least one microcontroller; modern cars are embedded with many microcontrollers in different subsystems; and mice are also embedded with microcontrollers.

Many low-cost microcontrollers are commercially available. Texas Instruments (TI) MSP430 and Atmel Atmega328 are two examples. Their parameters are listed in Table 2.2. As we can see, their computation capabilities are limited, thus we should use expensive algorithms only when necessary.

Memory An important component is memory. Memory categories and usages are shown in Table 2.3. Intermediate data is stored in random access memory (RAM). RAM loses

[1]http://www.capsil.org/capsilwiki/index.php/TELOSB/TMote_Sky, access time 2013-11-2
[2]http://arduino.cc/en/Main/ArduinoBoardUno, access time 2013-11-2
[3]http://arduino.cc/en/Main/ArduinoBoardMega2560, access time 2013-11-2
[4]http://arduino.cc/en/Main/ArduinoBoardLilyPad, access time 2013-11-2

Table 2.2: Examples of microcontroller parameters

	TI MSP430	Atmel Atmega328
Architecture	RISC	RISC
Operating frequency	25 MHz	20 MHz
Word length	16-bit	8-bit
Memory size	Varying	32 KB flash memory, 1 KB EEPROM

Table 2.3: Memory category and usage

Category	Program memory	RAM
	ROM, flash memory, EEPROM	
Usage	Store program code	Store intermediate data

Table 2.4: Examples of transceiver parameters

Transceiver	Standard	Band	Rate	Range
XBee ZB	ZigBee	2.4 GHz	250 kbps	120 m
XBee-PRO 802.15.4	IEEE 802.15.4	2.4 GHz	250 kbps	3.2 km
CC2420	IEEE 802.15.4	2.4 GHz	250 kbps	N/A

data if the power is off, thus a piece of program code is stored in program memories, for example read only memory (ROM), flash memory, and electrically erasable programmable read only memory (EEPROM).

Nodes generally have only limited memories, thus it is necessary to optimize the memory usages. In our case, we need to achieve a balance between applications and security mechanisms. This means that we cannot store a large amount of code for security mechanisms, or a large number of cryptographic keys.

Radio transceiver The radio transceiver is a combination of a transmitter and a receiver. It is in charge of sending and receiving wireless radio signals. XBee ZB[5], XBee-PRO 802.15.4[6] and Chipcon CC2420[7] are three examples of commercially available low-cost transceivers. Their parameters are listed in Table 2.4.

The radio transceiver is the most commonly used data communication device. This is because it has the following advantages: relatively long range, high data rates, acceptable error rates at reasonable energy expenditure, and it does not require a line-of-sight between

[5]http://www.digi.com/xbee/, access time 2013-11-2
[6]http://www.digi.com/xbee/, access time 2013-11-2
[7]http://www.ti.com/product/cc2420, access time 2013-11-2

transmitter and receiver.

Sensors and actuators Sensors and actuators are the key components. We list some commercially available sensors and actuators in Table 2.5; and they are introduced below.

Physiology sensors measure physiological states. The available sensors include electrocardiography (ECG) sensors, electroencephalography (EEG) sensors, electromyography (EMG) sensors, blood glucose sensors, pulse oximeters, and blood pressure sensors. They are useful in most medical applications.

Kinematic sensors provide a measurement of human movement, absolute geographic position, velocity, and acceleration. The available sensors include accelerometers and gyroscopes. These sensors are useful in applications such as sports training, interactive gaming, and context-aware services.

Behavioural sensors are used for voice recognition, facial pattern recognition, and sweat measurement, describing human moods and emotions. The available sensors include cameras, microphones, and humidity sensors. They are widely used in current mobile phones, and are foundations of affective computing applications.

Environmental sensors detect and characterize the surrounding environment of users. The possible information includes pollution, water or airborne allergens, and ultraviolet light intensity. These sensors are also useful for those people who are allergic to pollen.

Actuators include LEDs and buzzers, which can be found in most current nodes. In addition, some medical implants, for example pacemakers and cochlear implants, should also be categorized as actuators.

Energy source and energy-saving designs There are normally three energy sources: battery, energy scavenging, and wireless energy transmission. Batteries are the most common power sources. In recent years, researchers have designed many ways of prolonging the lifetime of a node by scavenging energy, for example the energy is generated using body movements, body temperature differences, and solar cells [89, 24]. In addition to energy scavenging, contactless energy transmission over a short range is also a good option for wearable and implanted nodes [55].

Table 2.5: Sensors and actuators

Sensor and Actuator		Usage
Physiology	Electrocardiography (ECG)	Monitors electrical activity of heart, and health status
	Electroencephalography (EEG)	Monitors electrical activity within the brain
	Electromyography (EMG)	Monitors electrical signals produced by muscles
	Blood glucose	Monitors blood sugar
	Pulse oximeter	Monitors oxygen saturation
	Blood pressure sensor	Monitors blood pressure
Kinematic	Accelerometer	Recognizes body posture, virtual reality, sports, and electronic games
	Gyroscopes	Detects orientation, movement monitoring
Behavioural		Monitors human mood and emotion
Environmental		Monitors surrounding environment parameters
Buzzer		Generates sound
LED		Generates light
Pacemaker		Regulates heartbeat rate
Cochlear implant		Restores hearing

Low-power designs are also important to the lifetime of nodes. The starting point is low-power circuit and system design, for example, better microcontrollers that can sleep in order to reduce energy consumption. In the next step, dynamic power management techniques play an important role. For example, when nodes are not assigned with tasks, these techniques allow microcontrollers to stay in low-energy consumption states, e.g. the sleep state.

2.2.2 Software Platform

Operating systems for dedicated nodes are basic software platforms that provide hardware and networking abstractions of nodes. They are less complex than general-purpose operating systems. Firstly, personal networks are typically deployed for particular applications, thus their operating systems can be optimized. Secondly, it is impossible to run complex systems on the resource-constrained nodes. TinyOS and Contiki are two famous operating systems.

Example (TinyOS). TinyOS [67] is a widely used event-driven operating system. When

an external event occurs, TinyOS signals the appropriate event handler to handle the event. It aims to support resource-constrained hardware platforms, e.g. Berkeley motes and Tmote Sky. □

Example (Contiki). Contiki [38] is an operating system which uses a simpler programming style in C. It is an open-source operating system, which allows tiny low-power systems to communicate with the Internet. □

In addition to operation systems, there are also many software libraries that provide interfaces to programmers. For example, nesC is a language used in TinyOS to implement TinyOS components and applications.

2.3 Communication Technologies

In this section, we will introduce communication technologies using a layer-based approach. The subsections are (1) design criteria, (2) physical layer, (3) media access control sub-layer, and (4) network layer.

2.3.1 Design Criteria

The design criteria [59, 107, 65, 28] of communication technologies can be summarized as follows:

- Energy efficiency. This is important because the lifetime of a node can be extended; this is critical for some implanted nodes, because replacement or recharging leads to a cost and convenience penalty. In addition, less heat is generated by nodes, and is absorbed by human tissue. Besides, node can be smaller and lighter, because it uses a smaller battery.
- High performance. This refers to maximum throughput and minimum delay.
- Quality of service (QoS). This means that data is received correctly in a reasonable period of time.
- Robust. Personal networks need to be robust against frequently changing topologies, caused by movements of the human body.

- Adapts to low redundancy. The group sizes are small in order to make users feel comfortable, thus in most cases there are no redundant nodes.
- Adapts to heterogeneity. Heterogeneity includes different data rates, mobility patterns, and QoS requirements.

2.3.2 Physical Layer

The physical layer consists of the networking hardware transmission technologies. These technologies translate logical signals to physical signals; and they transmit the physical signals over physical links.

Radio is the widely used physical layer technology in personal networks. A radio channel uses radio waves as the transmission medium. This medium has the following advantages: easy to generate, travels long distances, penetrates walls, and travels in all directions.

Free space radio channels Free space radio channels around the human body exist in two situations: line-of-sight and non-line-of-sight. In the line-of-sight case, the transmitter and receiver are located at one side of the body. In the non-line-of-sight case, there is no direct view between the transmitter and receiver. Some related results [107, 65, 28] are described below.

Firstly, non-line-of-sight radio channels do exist, because radio waves are more likely to diffract around the body rather than have a direct path through the body. For example, the dominant radio channel between two ears is the non-line-of-sight radio channel; the straight-line transmission from one ear to the opposite one can be neglected.

Secondly, a higher path loss along the non-line-of-sight channel than along the line-of-sight channel was observed. There are two reasons: radio waves diffract around the human body; and the human body absorbs a larger amount of radiation.

Thirdly, body movements, body shapes, and node positions can influence radio signals. For example, significant attenuation can occur when the line-of-sight link between two nodes is blocked by an arm.

Table 2.6: Media access control protocols

Protocol	Objective	Technique feature
BSN-MAC [68]	Energy efficiency	Feedback mechanism
H-MAC [69]	Energy efficiency	Mechanisms based on heartbeat rhythm
B-TDMA [101]	Energy efficiency	Mechanisms based on battery discharge dynamics, wireless channel models, packet queuing characteristics
CICADA [64]	Performance, energy efficiency mobility resilience	Cross layer design
BodyQoS [127]	QoS	Asymmetric QoS framework, virtual media access control, adaptive bandwidth scheduling
DQBAN [87]	QoS, energy efficiency	Cross-layer fuzzy rule-based scheduling algorithm

Radio channels inside the human body Some results regarding radio waves inside the human body have been summarized in [107, 65, 28]. There are two main results. The first result is that the radio waves are attenuated considerably before they reach the receiver. Thus the radio communication inside the human body requires more energy than in a free space.

The second result is that the nodes produce heat during communication. The heat is absorbed by the surrounding tissue, and causes a temperature increase of the tissue. Since the tissue is sensitive to temperature increase, the energy consumption should be restricted to a minimum.

2.3.3 Media Access Control Sub-Layer

The media access control sub-layer is the protocol layer that transfers data between neighbouring network nodes. It provides addressing and channel access control mechanisms.

Many low-power media access control protocols have been proposed for general wireless sensor networks; however, authors of [68] argue that these protocols are not suitable for personal networks with varying power requirements and traffic characteristics. Therefore, media access control protocols designed in particular for personal networks are still required. Some protocols are summarized in Table 2.6, and they are introduced below.

BSN-MAC [68] is an adaptive protocol designed for personal networks. Using feedback

information from slave nodes, the protocol adjusts parameters dynamically in order to save energy.

H-MAC [69] achieves time synchronization using heartbeat rhythm information. Energy consumption due to time synchronization mechanisms can be significantly reduced.

B-TDMA (battery-aware TDMA protocol) [101] takes into account the battery discharge dynamics, wireless channel models, and packet queuing characteristics. This protocol can increase the battery lifetime while satisfying performance and QoS requirements.

CICADA (cascading information retrieval by controlling access with distributed slot assignment) [64] is an efficient protocol that can be used in personal networks. CICADA offers low delay and good resilience to mobility. It is also energy-efficient because it supports sleep mechanisms.

BodyQoS [127] focuses on quality of service. It uses an asymmetric architecture, a virtual media access control protocol, and an adaptive resource scheduling mechanism.

DQBAN (distributed queuing body area network MAC protocol) [87] uses a cross-layer fuzzy-rule scheduling algorithm and energy-aware radio activation policies. It improves QoS and energy efficiency in personal networks that are used in healthcare scenarios.

2.3.4 Network Layer

Network layer protocols provide the means for transferring data sequences from a source to a destination via one or more networks. There are three main design strategies: multi-hop routing, heat-aware routing, and cross-layer design. Related protocols are summarized in Table 2.7, and they are introduced below.

Multi-hop routing Multi-hop routing is useful. First of all, multi-hop routing can be more energy-efficient [124, 19]. In addition, it is generally better in the following cases: slave nodes are far from the coordinator node, line-of-sight links are blocked, and direct communications are not possible. Also, multi-hop routing can be more reliable [78].

Heating-aware routing Heating-aware routing protocols are proposed in order to mitigate radiation absorption and heating effects. Firstly, in [93], the authors show that the

Table 2.7: Network layer protocols

Protocol	Objective	Technique
[124, 19]	Energy efficiency	Multi-hop networking
[78]	Reliability	Multi-hop networking
[93]	Reduce heat absorption	Power scheduling algorithms, traffic control algorithms
LTR, ALTR[12]	Reduce heat absorption	Chooses the neighbouring node with the lowest temperature
LTRT[103]	Reduce heat absorption	Combines node temperature and Dijkstras algorithm
CICADA [64]	Performance, energy efficiency mobility resilience	Cross layer design
[95]	Energy efficiency	Divides network into time zones
WASP[18]	High packet delivery radio, energy efficiency	Spanning tree, divides the time axis in slots

heating effects can be reduced by power scheduling and traffic control algorithms that balance the communications over the nodes. Secondly, in [12], LTR and ALTR are proposed. LTR chooses the neighbouring node with the lowest temperature as the next hop for routing. In addition, it discards packets that exceed the maximum hop count. ALTR is similar to LTR, but it use the shortest hop routing instead of discarding the packets. Finally, in [103], the Least Total Route Temperature (LTRT) that combines node temperature and the Dijkstras algorithm is designed. It achieves a lower hop count per packet, a lower number of packets dropped, and a lower temperature rise.

Cross-layer design Cross-layer design can improve the efficiency of the protocols by combining two or more layers. CICADA, introduced in the last subsection is a good example. In addition, Ruzelli et al. [95] claim that their cross-layer protocol built on IEEE 802.15.4 is useful in PNs. They divide the network into time zones; each time zone takes turns in the transmission. The protocol significantly extends the network lifetime. Also, the wireless autonomous spanning tree protocol (WASP) [18] sets up a spanning tree, and divides the time axis in slots. It achieves a high packet delivery ratio while keeping energy efficiency.

2.4 Applications

Personal networks enable many interesting applications, especially medical applications. In this section, we will review these.

2.4.1 Medical applications

We categorize medical applications as vital sign monitoring systems, human activity detection systems, medical implant applications and emergency response applications. These applications are introduced below.

Vital sign monitoring systems These systems usually use personal networks in order to obtain vital signs. Examples of vital signs are pulse rate, ECG, body temperature, and blood pressure.

Firstly, these systems can be used as health-care systems. Example systems are ALARM-NET [121], MobiHealth [108], HealthGear [85] and Mobile ECG [35].

Secondly, they can also be used for helping patients. For example, Ubimon [80] is used to manage patients with ischaemic and arrhythmic heart disease, and AWARENESS [116] is designed to detect and react to epileptic seizures.

Besides, they can be designed for specific users. For example, LifeGuard [77] is developed for astronauts, FireLine [13] is designed for fire fighters, and Baby Glove [13] provides health-care services for babies.

Human activity detection systems Personal networks can be used to detect human activities and postures. These networks usually include accelerometers, microphones and cameras. Typical usages are listed below.

One usage is fall detection for elderly people. Example systems are Smart Home Care Network [102] and the fall detection system in [92].

Another usage is posture detection in the patient recovery period. Example system is HipGuard [53], in which a few nodes are placed on waist, thighs, shins and feet. This system produces alarms when patient posture is not good for recovery.

Implant applications In a medical implant system, a person is implanted with several internal nodes, and these can interact with external nodes. Some examples [105] are as follows. In a cochlear implant system, the external node sends sound data to the internal node via wireless radio channels. Additionally, in a retinal implant system, the external video-capture unit sends image data to the implanted device via wireless channels. Also, Avery Biomedical has developed a breathing pacemaker system, in which an internal node can get power and signal via wireless radio channels; this is good for patients because there are no long-term issues with wound-care management, and no chronic infection risk since the patient's skin is intact.

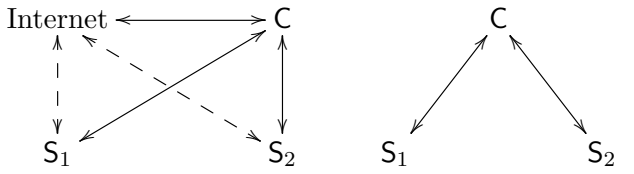
Emergency response applications There are many personal network-based emergency response systems. For example, CodeBlue [74] enables wireless monitoring and tracking of patients in a disaster response scenario; AID-N [41] aims at dealing with mass casualty incidents; and SMART [33] can monitor physiological signals from patients in the waiting areas of emergency departments.

2.4.2 Other applications

There are many other interesting applications, including context-aware computing, affective computing, sports training, and interactive gaming. They are introduced below.

Context-aware computing In a context-aware system, a person wears several slave nodes to measure location and other context information. Services are provided based on the context information. For example, an in-vehicle sensor network [97, 30, 56] collects facial expression and physiological signals in order to determine the alertness level of the driver; and it can send a warning to the driver based on the collected information. In addition, PrivacyIoT [51, 52], which integrates personal networks, home sensors and other environmental sensors, is a privacy-preserved context-aware service platform.

Affective computing In an affective computing system, personal networks monitor users' emotional reactions by, for example, heart-beat and respiration rates. We list several example systems as follows. PEACH [104] detects changes in patients' physiological and



(a) Internet of things (b) Autonomous network

Figure 2.1: Example application models

emotional states, and shares this information with interested people. DigitalBeing [39] provides a platform for dancers. Music and lighting in the dancing environment are adjusted, based on the dancers' feelings and moods.

Sports training and interactive gaming In sports training and interactive gaming systems, personal networks use accelerometers and other sensors to identify specific postures [32, 44, 45, 88]. There are many commercial products, for example, Nike + iPod sensor [5], Adidas miCoach Heart Rate Monitor for iPhone/iPod [2], and Wii Remote Plus controller [4] that can detect movement in three dimensions.

2.4.3 Application Models

There are generally two application models: Internet of things and autonomous network. These two network models are explained below.

Internet of things In this model, a coordinator node is connected to the Internet. Slave nodes are only connected to a coordinator node in most cases, but they can also be connected to the Internet depending on the requirements.

Internet of things is useful. It enables authorized practitioners to remotely access sensor data. Additionally, if any abnormalities are found, personal networks can send an alarm to remote practitioners. Besides, important sensor data can be stored in remote databases for further processing.

Example (Internet of things). In Fig. 2.1(a), we show an example application. It consists of a coordinator node C and two slave nodes S_1 and S_2. S_1 and S_2 are connected to C, and C is connected to the Internet; the connections are depicted as solid arrows. S_1 and S_2 can also be connected to the Internet; the connections are depicted as dash arrows. □

Autonomous network In autonomous networks, nodes are not connected to the Internet. For example, Wii Remote Plus controllers [4] are generally not connected to the Internet.

Example (Autonomous network). In Fig. 2.1(b), we show an example application. S_1 and S_2 are connected to C; the connections are depicted as solid arrows. There are no Internet connections. □

2.5 Security Threats

In this section, we will review security threats. It will include (1) active attacks; (2) passive attacks; and (3) physical attacks.

2.5.1 Active Attacks

The attacker A may initiate many active attacks based on for example spoofing, replay, and message modification. These attacks are introduced below.

In a Sybil attack [79], a single malicious node A illegitimately uses multiple identities. This makes other nodes believe in a wrong topology, which can significantly interrupt protocols in the media access control sub-layer and the routing layer. In Fig. 2.2(a), A uses fake identities: A_1 and A_2. In this case, A obtains an advantage in aggregation protocols.

In a hello flooding [60] attack, A broadcasts "Hello" packets with high transmission power; thus, other nodes think that A is their neighbour. In Fig. 2.2(b), A can make C, S_1, and S_2 think that A is their neighbour using high-power "Hello" messages.

In a wormhole [60, 114, 10] attack, A tunnels messages received in one part of the network and replays them in another part. This will disrupt the routing mechanisms.

In a sinkhole attack, A manipulates the neighbouring nodes to lure nearly all the traffic from a particular area through a compromised node. This malicious node can now not only tamper with the transmitted data but can also lead to other attacks like a black hole and a selective forwarding [60, 10].

In a black hole attack [10], A drops all the received packets. When A is also a sink hole, this attack is more effective. This attack interrupts the MAC and routing protocols.

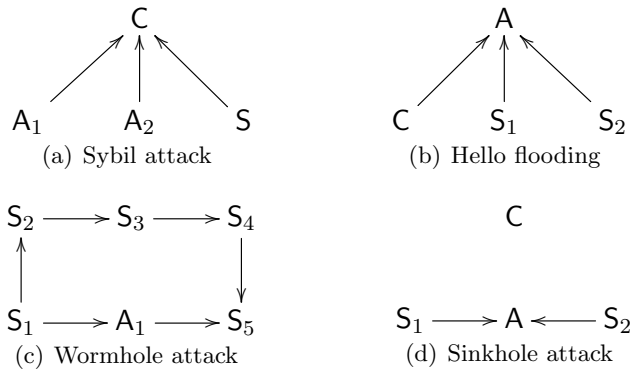


Figure 2.2: Attacks against personal networks

In a selective forwarding attack, a malicious node may refuse to forward certain messages and simply drop them, ensuring that they are not propagated toward their destination. If this node drops all the packets it receives then the neighbouring nodes will think the node is down and will look for an alternative route. The malicious node may also choose to selectively drop some messages but forward the remaining traffic [60].

The attacks against data aggregation protocols [11], which are used to aggregate sensor data in order to reduce communication overhead, are mainly as follows: false reports from compromised aggregation coordinator; false data from compromised slave node.

In attacks against time synchronization protocols, which are used to synchronize the time differences in different nodes, the main objective of an attacker is to deceive other nodes into thinking that an incorrect time is accurate [75]. These attacks can be separated as (1) internal attacks: the attacker, as a neighbouring node, lies about the value of its internal clock; and (2) external attacks: manipulation of the negotiation messages.

In denial of service (DoS) attacks, A makes the networks unavailable to the users. Targets of DoS attacks are mainly the nodes or the communication channels [120].

The first type of DoS mainly aims to deplete the battery life of nodes. Firstly, an attacker A can repeatedly send fake signatures to a node. The verication of a cryptographic signature is expensive, thus A can deplete the battery life of this node in a short time. Secondly, A can prevent nodes from entering sleeping mode. In this case, nodes run out of power in a short period.

The second type aims to block the wireless radio channels. A can constantly emit wire-

less radio signals that do not follow an underlying MAC protocol; thus, any member of the network in the affected area will neither be able to send nor receive any packet. Alternatively, A can choose some optimized versions: random jamming, in which the attacker alternates between sleep and jamming to save energy; reactive jamming, where the jam signal is transmitted only when the attacker senses traffic; and jamming based on MAC protocols, e.g. by jamming only request-to-send packets.

2.5.2 Passive Attacks

Eavesdropping Eavesdropping means that A is able to access the content of the transmitting messages. Eavesdropping attacks do not directly influence the behaviour of the network, and so are passive attacks.

Example. Suppose A can get message IDs, node IDs, and node location. A can deduce whether the user is at home or not. If A is a thief, A is able to choose a suitable time to break into the user's house. □

Example. Many cryptographic protocols for personal networks distribute security credentials as clear text in pre-distribution phases. Thus, it is possible for A to get these credentials, and compromise these protocols. □

Due to the broadcast nature of the wireless radio channel, personal networks are susceptible to eavesdropping attacks. However, the used wireless radio technologies generally only support short-range communications; thus, A must be close enough to initiate the attacks, which means that these networks are generally more secure than other long-range wireless networks. Nevertheless, this advantage cannot stop a determined attacker; thus, security protections are still necessary.

Traffic analysis Attacker A may monitor the network traffic in order to analyze the traffic pattern. This attack does not directly influence the behaviour of the network, and is a passive attack.

Traffic analysis is always the first step of other attacks. First of all, it helps A to get the network topology information. Secondly, analyzing a single node is also meaningful. For example, coordinator nodes are always busier than other nodes; A has a high probability

of compromising the whole network by compromising the busiest node. Thirdly, the traffic analysis may lead to privacy leakage, for example information regarding user locations and habits.

Example (Privacy leakage). A user wears a personal network that submits sensor data to remote applications via some fixed gateway nodes. When a fixed node located in a bedroom is activated, A can deduce that the user is in the bedroom; when a fixed node in a washroom is activated for a long time, A can deduce that the user may be taking a bath. □

2.5.3 Physical Attacks

Physical attacks mean that A is able to capture some nodes, or to initiate side-channel attacks.

Node capture If A captures one node, A can launch many attacks, for example accessing the memory and injecting malicious code. As a result, A can obtain keys and sensitive sensor data stored in the node; A can modify the sensor data; A can selectively report the sensor data; and A can forge messages that appear to be from a legitimate node.

Side-channel attacks In side-channel attacks, A monitors certain physical properties of the nodes in order to deduce useful information. Side-channel attacks include power analysis, electromagnetic attacks, and timing attacks [94, 84].

In power analysis attacks, A studies the power consumption of devices, especially the energy used by cryptographic operations. A can find features by analyzing single power traces or a large number of power traces.

In electromagnetic attacks, A derives the power traces from the electromagnetic interaction phenomenon that is collected by electromagnetic probes. A then analyzes the power traces. A can also use more advanced techniques, for example adding spatial information to the measurement data or analysing the frequency domain.

In timing attacks, A exploits the execution time variance of different cryptographic system branches. These attacks can succeed because the execution time of cryptographic algorithms is often different, depending on the inputs.

2.6 Security mechanisms

In order to mitigate security threats, various security mechanisms have been proposed. In this section, we will review (1) design criteria, (2) related definitions, (3) requirements, (4) pre-distribution-based mechanisms, (5) physiological value-based mechanisms, (6) asymmetric key-based mechanisms, and (7) physical layer-based mechanisms.

2.6.1 Design Criteria

The design criteria of security mechanisms used in personal networks can be summarized as follows:

- Energy efficiency. This refers to communication and computation efficiency. Generally speaking, wireless radio communications consume more energy than computation; thus, reducing wireless radio communications is more important;
- Memory efficiency. This is important; however, since group sizes are generally small, it is not critical.
- Robust against frequently changing topology if the security mechanisms operate continuously.

The probability that attackers physically capture nodes can be neglected, because the nodes are worn by or implanted in human users. In addition, for cryptography keys, the metrics are listed as follows:

- Flexibility. This should be easy to add node, delete node, and update keys; this is significant.
- Storage. It refers to the memory required for storing cryptographic keys; since group sizes are generally small, it is thus important but not critical.
- Resilience of key compromise. This means that the whole network should not be compromised when some nodes are compromised. Because we can reasonably assume that capturing a node physically will be noticed and prevented by users, this is not critical.

- Connectivity. This refers to the minimum number of secure links that are required to be removed in order to partition a network; it is an important measure of network robustness.

2.6.2 Definitions of Key Establishment Protocols

The key establishment protocol is defined as Definition 2.1. In addition, key agreement protocol, key transport protocol, and hybrid key establishment protocol are defined based on Definition 2.1. Details of these definitions can be found in [17].

Definition 2.1 (Key Establishment Protocol). A key establishment protocol is a prescribed sequence of interactions between entities; these entities establish a session key in order to secure their subsequent communications using cryptography.

Definition 2.2 (Key Agreement Protocol). A key agreement protocol is a key establishment protocol where all the parties influence the generation of session keys.

Definition 2.3 (Key Transport Protocol). A key transport protocol is a key establishment protocol where only one party influences the generation of session keys.

Definition 2.4 (Hybrid Key Establishment Protocol). A hybrid key establishment protocol is a key establishment protocol where several parties (more than one, but not all of the parties) influence the generation of session keys.

2.6.3 Requirements of Key Establishment Protocols

The requirements of key establishment protocols are generally authenticity, integrity, confidentiality and availability. Authenticity is the first fundamental requirement. This requirement can be further elaborated as follows:

- Authentication of origin: a message that is claimed from a certain node is indeed originated from that node.
- Entity authentication of S to C: node C is assured that S is involved in a protocol, and S has actually participated.

- Mutual entity authentication: both entities C and S are authenticated to each other in the same protocol.

Secondly, integrity states that the key has not been modified by the adversary or, equivalently, has inputs only from legitimate nodes.

- For a key transport protocol, key integrity means that if a key is accepted by any node, it must be the same key as chosen by the key originator.
- For a key agreement protocol, key integrity means that if a key is accepted by any node, it must be a known-function output of the inputs from legitimate nodes.

Thirdly, confidentiality means that the session key should only be understood by the legitimate nodes. Finally, availability requires that the protocol can provide services whenever the user requests. The denial-of-service attacks mentioned previously should be prevented to the largest extent.

2.6.4 Pre-distribution-Based Mechanisms

In the past decade, many security protocols proposed for personal networks are based on a pre-deployment phase. Generally speaking, the procedure is as follows: (1) secrets are deployed in a pre-deployment phase; and (2) symmetric keys and IDs are securely distributed using the secrets. However, there are some problems. For example, it is always difficult to update pre-deployed secrets even when they are compromised.

Reducing the number of keys Reducing the number of keys is the main objective. However, this is usually at the cost of connectivity, which is not good for personal networks since there are not many redundant links. In addition, removing a node may significantly reduce the connectivity in some cases.

The fundamental work is a random key pre-distribution scheme proposed by Eschenauer and Gligor (E-G scheme) [40]. In the first step, a key pool is generated. The key pool contains a large number of keys, say p keys. Each node randomly selects k ($k << P$) different keys from the key pool. In the second step, if two neighbour nodes have the same keys, they can establish a secure link with each other directly. The probability is

$1 - \frac{((p-k)!)^2}{(p-2k)!p!}$. In the third step, all nodes establish secure links via the links established in the second step. The first advantage of the E-G scheme is that each node only needs to store a small number of keys. The second advantage is that network deployment information and a central trusted station are not required.

Some researchers focus on improving the resilience to node compromise in the E-G scheme. Chan, Perrig and Song [26] suggest a q-composite scheme, in which neighbour nodes establish a secure link only if they share at least q keys. In addition, Du, et al.[37] suggest a combination of the E-G scheme and the Blom scheme [15]; and Liu, et al. suggest a combination of the E-G scheme and the Blundo scheme [16].

Key distribution can use deployment knowledge to achieve better performance. For example, Du, et al. [36] divide the key pool into smaller ones corresponding to the geographic node groups, and the nearby key pools share more keys. Huang, et al. [49] also propose a structured key pool instead of the large key pool. This scheme pre-distributes keys within a geographic zone and between two adjacent zones. In the GBKP scheme [71] and the PIKE scheme [25], nodes are allocated into grids. Each row or column is associated with one polynomial. After a procedure similar to the Blundo scheme [16], nodes in the same row or column have shared keys. Using these keys, nodes in different rows and columns can also establish shared keys. These schemes avoid unnecessary keys based on the deployment information, which reduces the storage consumption, but their deployments are not flexible.

Except for the E-G scheme family, the balanced incomplete block design (BIBD) is another optimization method [21, 66]. A (p, k, q)-BIBD allows each node to pre-load k keys from a pool that contains p keys. Every pair of nodes has q common keys. Typically, a $(gs^2 + gs + 1, gs + 1, 1)$-BIBD is used, where gs is the number of nodes. In paper [96], the BIBD design is combined with the Blundo scheme, in the sense that each slave node is preloaded with polynomials.

Comments. There are some disadvantages. For example, it is difficult to update pre-deployed secrets (even in the event that they are compromised, or suspected of having been compromised). In addition, it is difficult to add new nodes or remove nodes. Besides, when several users are involved, for example patients and doctors, they have to make the pre-deployed secrets public, which is not preferred by many users.

2.6.5 Physiological Value-Based Mechanisms

In [29], Cherukuri et al. establish secure communications using a physiological value. They also introduce an error correction approach in order to remove the variance of values that are measured at different points on the body. In [112, 111, 113], Venkatasubramanian et al. provide security in personal networks using physiological value.

Comments. The security of these mechanisms is doubtful when contact-free measurement devices are available. For example, ECG information can be measured remotely [91, 23, 42, 123]. Thus, cryptographic keys, which are generated using ECG data, can be generated by attackers.

2.6.6 Asymmetric Key-Based Mechanisms

Asymmetric key-based schemes use a pair of public and private keys to initiate secure connections. Computation and memory efficiency are the main issues.

TinyPK [115] uses RSA-based certificates to authenticate external parties before they can access the network. This scheme chooses a low exponent RSA system in order to reduce the computation cost. The scheme can further reduce the cost by placing the computationally expensive operations on the powerful external parties.

In addition, since elliptic curve cryptography is generally more efficient regarding the memory consumption, Liu and Ning have designed a software library TinyECC [70]. It provides a digital signature scheme ECDSA[8], a key exchange protocol ECDH[9], and a public key encryption scheme ECIES[10]. Huang et al. [50] further reduce the computation overhead on slave nodes by putting expensive asymmetric key operations on the security managers.

Zhang et al. [125, 126] have proposed a location-based approach. The core concept is that private keys are binding with their IDs and locations. This solution has perfect resilience to node compromise, as long as the location information is trustworthy.

Comments. Key establishment protocols using asymmetric key algorithms in personal networks have not been studied throughly. The location-based approach depends on the

[8] Elliptic Curve Digital Signature Algorithm
[9] Elliptic Curve Diffie-Hellman
[10] Elliptic Curve Integrated Encryption Scheme

location information, which makes it only suitable for static networks. ECDH is an unauthenticated key establishment protocol, which will be improved in Chapter 4.

2.6.7 Physical Layer-Based Mechanisms

Security mechanisms in the physical layer have been neglected in most cases. However, there have been some interesting results in recent years.

The fundamental principle of a secrecy system was formalized by Shannon in [99]. After this, Wyner [122] introduces the wiretap channel that makes secrecy achievable between a pair of legitimate transmitter and receiver even when there are eavesdroppers. It is mainly based on the assumption that the channel from transmitter to eavesdropper is "noisier". Recently, the wiretap channel has drawn much attention, and most of these studies are based on multiple antennas [61, 62, 83, 98, 72].

In addition, a secret key between two wireless devices can be extracted using received a signal strength indicator [76]. This method has been evaluated in actual environments [54].

Comments. These mechanisms are based on the difference of signal-noise ratios (between the receiver and the eavesdropper). Firstly, the assumption– the channel from transmitter to eavesdropper is "noisier"– is not always true. Secondly, the key generation rate is limited by the conditions of wireless radio channels; the key generation may need a long time.

2.7 Standards

In this section, we will introduce several related standards: IEEE 802.15.4, Zigbee, Bluetooth, and 6LoWPAN.

2.7.1 IEEE 802.15.4

IEEE 802.15.4 standard [8] specifies the physical layer, the media access control sub-layer, and corresponding security services in personal networks.

Physical layer The physical layer (PHY) of IEEE 802.15.4 standard specifies data transmission management services using wireless radio transceivers. In 2003, the standard in-

Figure 2.3: One example IEEE 802.15.4 PHY packet [8]

cluded only two regional bands: 902 to 928MHz and 868 to 868.6MHz, and one worldwide band: 2.4 GHz. In the following years, new frequency bands and techniques have also been included.

One typical PHY packet is shown in Fig. 2.3. Preamble is 32 bits; it is used for symbol synchronization. The packet delimiter is 8 bits, which is used for frame synchronization. PHY header is 8 bits, it includes the data length. PSDU is the data field, and it is up to 127 bytes.

Media access control sub-layer The media access control (MAC) sub-layer enables transmission of media access control frames between neighbouring nodes. Firstly, the standard defines two kinds of node:

- Full-function device (FFD). It can be used as the coordinator in a personal network; usually, at least one FFD is required in the network.
- Reduced-function device (RFD). It provides only simple functions; RFDs usually work as leaf nodes.

Secondly, the standard supports peer-to-peer and star topology. Besides, frame structures are defined. There are four basic frames: beacon frame, data frame, acknowledgement frame, and MAC command frame. The standard also defines a superframe that supports sleeping mechanisms. One typical frame is shown in Fig. 2.4.

Security IEEE 802.15.4 standard supports data confidentiality, data authenticity and replay protection. These mechanisms are mainly specified in the media access control sub-layer. For example, the Auxiliary Security field in Fig. 2.4 specifies the information of keys, the counter to prevent replay attacks, and the protection method chosen from Table 2.8. In addition, the data can be protected using AES and AES-CBC-MAC.

Each 802.15.4 node has an access control list (ACL). For example, when nodes C and S want to communicate with each other, C and S will look up their ACLs. If C is in the ACL

Frame Control	Seq Number	Destination Address	Source Address	Auxiliary Security	Data Payload	CRC

Figure 2.4: One example frame in IEEE 802.15.4 media access control sub-layer [8]

Table 2.8: Security protection methods [8]

Security attributes	Data confidentiality	Data authenticity	Authentication tag length (octets)
None	No	No	0
MIC-32	No	Yes	4
MIC-64	No	Yes	8
MIC-128	No	Yes	16
ENC	Yes	No	4
ENC-MIC-32	Yes	Yes	4
ENC-MIC-64	Yes	Yes	8
ENC-MIC-128	Yes	Yes	16

<table>
<tr><td rowspan="2">ZSEC</td><td>Zigbee APS, ZDO, AF</td><td rowspan="2">ZDO management</td></tr>
<tr><td>Zigbee NWK</td></tr>
<tr><td colspan="3">IEEE 802.15.4 medium access control sub-layer</td></tr>
<tr><td colspan="3">IEEE 802.15.4 physical layer</td></tr>
</table>

Figure 2.5: Zigbee stack architecture

of S and vice versa, C and S can communicate with each other using the specified security suite; otherwise, the communication is denied, or an authentication process is called.

2.7.2 ZigBee

ZigBee standard [7] specifies the application layer, network layer (NWK), and corresponding security services in personal networks. It defines a series of modules: an application support sub-layer (APS), a ZigBee device objects (ZDO), an application framework (AF), a ZigBee device profile (ZDP), and a ZigBee security service (ZSEC). The stack architecture is shown in Fig. 2.5.

Application layer The application layer includes AF, APS, ZDO. AF is the environment of application objects. APS is the interface between the network layer and the application layer. ZDO is the interface among application objects, the device profile and APS.

Network layer The ZigBee network layer is an intermediate layer between the IEEE 802.15.4 media access control sub-layer and the ZigBee application layer. The network layer ensures that the media access control sub-layer operates correctly; and it provides services to the application layer. We outline the key points below.

There are three types of devices: coordinator, router, and end device. The coordinator is the most powerful node; it manages the whole network. The router relays data from other nodes, and is relatively powerful. The end device normally has limited functionalities; it communicates only with its parent nodes.

Three topologies are supported. The first one is the star network. This network consists of a coordinator and end devices. The coordinator initiates and maintains all the end devices. The second is the tree network. This network consists of a coordinator, routers and end devices. This network is controlled by the coordinator and routers. Firstly, the coordinator initiates the network and chooses network parameters. Secondly, routers pass messages using a hierarchical routing strategy between the coordinator and end devices. The third is the mesh network. This network also consists of a coordinator, routers and end devices. Firstly, the coordinator initiates the network and chooses parameters. Secondly, the network provides peer-to-peer communications using ad-hoc routing protocols with the help of routers.

Data and management services are provided. A network layer data entity enables the transmission of NWK protocol data units. Meanwhile, a network layer management entity supports management services, for example new device configuration, network initiation, addressing and routing.

Security Security is an important part of the ZigBee standard. First of all, the usage of keys and their distribution methods are specified:

- Master key. This is a symmetric key for key distribution protocols. It can be pre-distributed or be sent via out-of-band channels.
- Link key. This is a shared key between two nodes for secure unicast communications. Zigbee suggests that link keys should be established using the master key.

- Network key. This is a global key for secure broadcast communications. This key is usually transported from the trust centre.

In addition, ZigBee provides security mechanisms at the network layer and application layer. Besides, ZDO manages security policies and device configurations.

Zigbee security services are highly relied on the trust centre, which is a trustworthy node. This node distributes keys to others. It can operate in commercial mode, which provides high security for commercial applications. It can also be in residential mode, which is designed for low-security residential applications.

2.7.3 Bluetooth

Bluetooth [3], especially Bluetooth low energy (BLE), is used to rapidly establish simple and short-range connections between low power devices and mobile terminals. The BLE channels are expected to provide a data rate up to 1 Mbps using 2.4 GHz band. BLE functionalities can be integrated into an existing classic Bluetooth controller, which is named as dual-mode; these functionalities can also be implemented independently with better performance, which is named single-mode.

Bluetooth specifies a security architecture that provides authentication, key derivation and confidentiality with custom algorithms based on the SAFER+ block cipher. Bluetooth authentication and key generation generally use PINs that are imputed in both devices. For devices without input interface, fixed PINs or secure simple pairing can be used. Secure simple pairing uses public key cryptography; and it either asks users to compare numbers, or use out-of-band communications, for example near field communication.

2.7.4 6LoWPAN

6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) [1] has defined encapsulation and header compression mechanisms that allow IPv6 packets to be sent to and received from over IEEE 802.15.4 networks. Fig. 2.6 shows the IPv6 protocol stack with 6LoWPAN. A simple IPv6 protocol stack with 6LoWPAN is almost identical to a normal IP stack, with the following differences: an adaptation layer LoWPAN is usually used to sup-

Application protocols
UDP, ICMPv6
IPv6
LoWPAN
IEEE 802.15.4 media access control sub-layer
IEEE 802.15.4 physical layer

Figure 2.6: 6LoWPAN protocol stack

port IPv6 over IEEE802.15.4; the most common transport protocol is the user datagram protocol (UDP); the Internet control message protocol v6 (ICMPv6) is used for control messaging; application protocols are often specifically designed.

The security of 6LoWPAN networks mainly relies on IEEE 802.15.4 security mechanisms. However, for powerful nodes, security mechanisms, for example IPsec and TLS, can also be used.

2.8 Chapter Summary

There are many different platforms and communication technologies for personal networks, each with their own characteristics. There is not one single perfect solution for all applications, thus trade-offs are made in different scenarios. However, in most cases, the designers should take into account the resource constraints, for instance, the constraint of energy, memory, and computation capability. Meanwhile, they are required to balance for example network performance, QoS, and robustness.

As personal network applications get more and more popular, security becomes one of the most pressing concerns. However, designing security mechanisms is a challenging task: the constrained resources make it very difficult to deploy strong security mechanisms. In order to meet the constraints, many innovative security mechanisms have been designed, for example, key pre-distribution-based mechanisms, physiological value-based mechanism, asymmetric key-based mechanisms, and physical layer-based mechanisms. Since they all have their problems, designing of suitable security mechanisms for personal networks is still an open issue.

In the following chapters, we aim to provide security mechanisms in personal networks.

These mechanisms should meet the following requirements: not dependent on a trusted third party; the identities and security credentials should be easily updated after deployment; they should be efficient; user burden should be minimized.

Chapter 3

Human Interactive Security Protocols

Human interactive security protocols (HISPs) are designed for providing security in wearable personal networks using human-controlled channels (HCCs). The concept design is as follows. In wearable personal networks, wireless radio channels are used for data communications, and properly selected HCCs are used to protect the data.

The main advantage of HISPs is flexibility. Firstly, HISPs do not need pre-distributed secrets, e.g. PIN code. Thus bootstrapping security in wearable personal networks is convenient, especially when the nodes do not belong to the same person, for example patients' nodes and doctors' devices in the waiting areas of emergency departments. Secondly, whenever the secrets are compromised, users can easily update them using HISPs.

Contents

3.1 Human-Node Interactions

This section includes (1) human interface devices; and (2) human-node interactions.

3.1.1 Human Interface Devices

Overview *Human interface devices* in nodes are devices that are embedded on nodes, and human users can interact with the nodes using these devices. Some examples of human interface devices and their capabilities are listed in Table 3.1. They are very different from those on computers: usually no keyboard, disk, or printer, and even no ordinary human interaction devices of any kind.

Whether certain interfaces are usable or not is largely dependent on the application scenarios. The first example: when C is a laptop, is generally not suitable for using an accelerometer as an input device. The second example: if the slave node is very small, display and LED matrix are not good choices.

Table 3.1: Human interface devices on nodes

Human interface device	Category	Capability
Small display	Output	Show short strings, short numbers, small pictures
Seven segment display	Output	Show numbers
LEDs (including bar graph and matrix)	Output	Flash with different patterns
Buzzer	Output	Beep with different patterns
Button	Input	Input 1-bit information
Accelerometer	Input	Get input by shaking the node
Microphone	Input	Detect sound
Vibration sensor	Input	Detect vibrations

Test bed We have developed a test bed node, shown in Fig. 3.1. It comprises a microprocessor, a radio transceiver, a small colour OLED display, five LED lights, a buzzer, and a user button. The output devices include the display, LEDs, and buzzer. The input device is the button. This test bed is a good example of understanding human interface devices on a typical node.

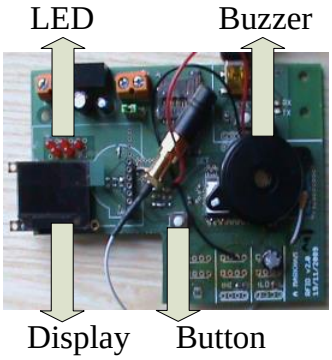


Figure 3.1: Test bed node.

3.1.2 Human-Node Interactions

Human-node interactions are interactions between human users and nodes using human interface devices. Thanks to these devices, although nodes are usually small and therefore do not support regular human-machine interactions, there are still surprisingly many interaction methods.

Human capabilities and human-node interactions In addition to human interface devices, human capabilities are also the foundation of human-node interactions. They can generally be separated into perceptions and actions, introduced below.

Perception allows human users to capture the output of human interface devices. Vision and hearing are the dominant perceptions. Vision is the capability of focusing and detecting images of visible light. Human users can sense the brightness and colour of a object. Hearing is the ability to perceive sound by detecting vibrations. Human users can sense the loudness and pitch of the sound. Typical interaction signals include strings, numbers, pictures, LED flashes and buzzer beeps.

Action allows human users to input signals to nodes via human interface devices. Typical actions include button-pushing and node-shaking.

Short review The papers [63, 57, 58] reported experiments regarding the usability of several human-node interaction methods. The related methods are (1) image comparison, (2) number comparison, (3) phrase comparison, (4) LED-button, (5) vibration-button, (6)

beep-blink, (7) blink-blink, (8) button-button. Their results are summarized below.

The first experiment is on the time of transferring a certain number of bits. These methods are categorized as follows:

- 0s -15s: image comparison, number comparison, phrase comparison
- 15s -30s: beep-blink, blink-blink
- 30s or more: LED-button, vibration-button, button-button

The second experiment is on ease of use. These interaction methods are categorized as three levels:

- Very easy: image comparison, number comparison, phrase comparison
- Easy: beep-blink, led-button, vibrate-button
- Other levels: blink-blink, button-button

The third experiment is on user preference. This experiment firstly separates the interactions into three groups GA, GB, and GC. Then, these interaction methods are compared within each group:

- GA (require displays on both nodes): image comparison and number comparison are better than phrase comparison
- GB (one node is interface constraint): vibrate-button is better than LED-button; and LED-button is better than button-button
- GC (synchronization comparison): blink-blink is better than beep-blink

The fourth experiment is on the error rates. Most methods perform well, reporting no errors or low rates around 5%. However, blink-blink, beep-blink, and image comparison show high error rates.

In summary, display-based methods, especially number comparison, are the best choices: (1) they require less time, (2) users feel easy using them, and (3) they have low error rates. When at least one node is interface-constrained, except the button-button method, all other methods (LED-button, vibration-button, beep-blink, and blink-blink) have their advantages.

3.2 Human-Controlled OOB Channels

This section includes (1) human-controlled channels; (2) their security models; and (3) their networking properties.

3.2.1 Human-Controlled Channels

Overview Human-controlled channels are established based on human-node or human-human interactions. Human-node interactions were introduced in the last section. *Human-human interactions* are interactions among human users. Generally speaking, the signals of human-human interactions are strings, numbers, pictures and conversations. In summary, we can define the human-controlled channel as follows.

Definition 3.1 (Human-controlled channel (HCC)). The human-controlled channel is a channel based on human-node or/and human-human interactions. Control has the following meanings: (1) users can determine which node will participate in a certain communication procedure; (2) users can determine when the node will participate in a certain communication procedure; (3) users can determine when the information will be transmitted; and (4) users can perceive and prevent interference during the communication.

We mainly consider the single-user scenarios, because only one user is involved in one personal network in most cases. Certainly, human-human interaction also exists. For example, a patient's nodes and a doctor's devices may need to work cooperatively in the hospital; and they tell each other the PIN code in order to set up the network. In this scenario, the conversation between patient and doctor is the human-human interaction.

Demonstration The available human-controlled channels using our test bed are display-human channels, LED-human channels, buzzer-human channels, and human-button channels. They use only existing human interactive devices on the test bed. In addition, they are controlled by human users. Their details are explained below.

Display-human channels use the small display as transmitter; and the receiver is the human eye. Users can compare strings or digits that are shown on displays; in some cases, a series of patterns shown on the displays can be compared as a whole, or individually.

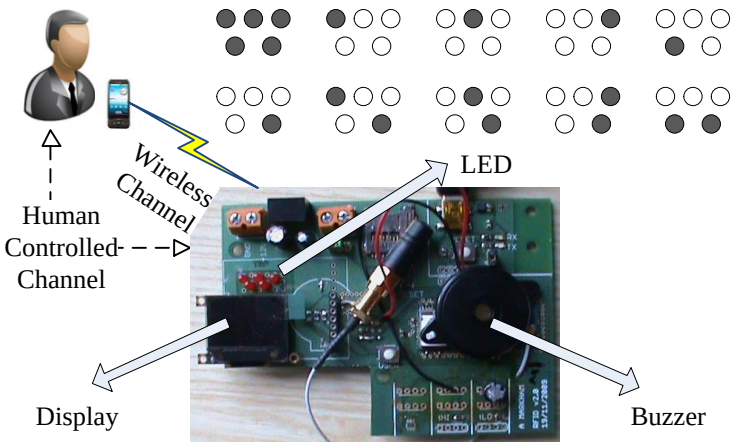


Figure 3.2: HCC demos.

LED-human channels use the LEDs as transmitter, and the human eye as receiver. There are several possible usages: (1) two nodes blink their LEDs, the user compares their blinking patterns; (2) LED combinations represent different numbers; for example, numbers from 0 (upper left) to 9 (lower right) are shown using LED combinations in Fig. 3.2; and (3) LED flashes as a 1-bit signal.

Buzzer-human channels use the buzzer as transmitter, and the human ear as receiver. There are several possible usages: (1) user compares the beep patterns; and (2) buzzer generates a beep as a 1-bit signal.

Human-button channels mean that human users push a button in order to transmit a 1-bit signal. This signal is usually used as confirmation. It is simple, but plays an important role in HISPs, as we will see later.

All these channels are human-controlled channels. The node is controlled by human users when worn by the users; in addition, users can perceive and prevent interference during the communication; for example, it is reasonable to assume that users will notice the physical attacks such as attacker A pushes a button on the legal node or manipulates the digits/string/patterns shown using displays and LEDs.

3.2.2 Channel Security Models

Channel categories From a security standpoint, channels can be categorized as follows [34, 31]: Dolev-Yao (DY), No-blocking (NB), No-spoofing (NS), No-spoofing NB (NSB),

Table 3.2: Attack models

Attack models	DY	NB	NS	NSB	CON	SEC
Overhear	Yes	Yes	Yes	Yes	No	No
Block	Yes	No	Yes	No	Yes	No
Modify	Yes	Yes	No	No	Yes	No
Fake	Yes	Yes	No	No	Yes	No
Replay	Yes	Yes	No	No	Yes	No
Delay	Yes	No	Yes	No	Yes	No

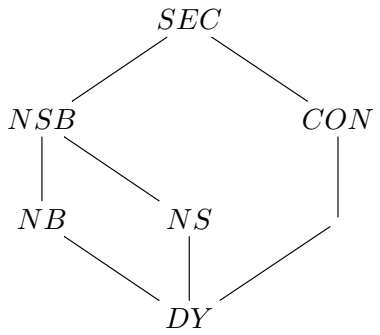


Figure 3.3: Channel security lattice

Confidential (CON), Secure (SEC). Categorization is based on what a computationally bounded adversary A can do to the messages over these channels, which are summarized in Table 3.2.

The relations among these channels are depicted in Fig. 3.3. The DY channel is the weakest channel; it is in the lowest level. NB, NS and CON channels are in the second lowest level. The NSB channel is higher. Finally, the SEC channel is the most secure.

NSB HCC One of the most useful channel models is a NSB HCC; we define this channel in Definition 3.2. In this dissertation, we mainly use NSB HCCs as the OOB channels in order to guarantee the authenticity, integrity and freshness of the data transmitted over insecure wireless radio channels. For convenience, we do not distinguish NSB HCCs from HCCs in the rest of this thesis.

Definition 3.2 (NSB HCC). The NSB HCC is a human-controlled channel, and should be modelled as a NSB channel from a security standpoint.

Threat model and assumptions Threat model. HCCs that will be used in our protocols are assumed to be NSB channels. In other words, attackers can overhear messages over HCCs, however they cannot block, modify, fake, replay, or delay messages. This threat model is based on assumptions elaborated below.

Assumption 1: users are trustworthy. It means that users will convey authentic messages among nodes without delay.

This assumption is reasonable. Because the application scenario is personal networks, users are generally the owners of these networks. In this case, we can reasonably assume that these users will not try to attack their own devices and networks.

Assumption 2: nodes are trustworthy. In other words, there are no malicious programs that will change messages shown using displays, LEDs, or other human interface devices.

Because the application scenario is personal networks, it is reasonable to assume that nodes have not been physically tampered by attackers. Also, users can regularly update systems or install anti-virus softwares, and thus prevent attacks against nodes. Therefore, we can reasonably assume that nodes are trustworthy.

Assumption 3: attacks that try to block, modify, fake, replay, or delay messages over HCCs can be found and prevented by users.

HCCs that we have implemented in Fig. 3.2, namely, display-human channels, LED-human channels, and human-button channels, are NSB channels. It is reasonable to assume that HCCs hold the following properties:

- Attacker A cannot directly manipulate the nodes, for example, push buttons on the nodes;
- A cannot manipulate the information shown using displays and LEDs.

Therefore, A cannot initiate fake conversations, block messages, replay messages, delay messages or modify messages over these channels.

Assumption 4: attackers can overhear messages over HCCs.

Attacker A can record the messages using for example cameras, and thus can easily overhear these messages. It means that HCCs are not confidential (CON) channels.

User expectations (1) Users will convey authentic messages among nodes without delay. (2) Users should update operating systems or even use anti-virus software in order to make sure that their nodes are trustworthy. (3) Users should not allow other people to use their nodes unless a) they are trustworthy, and b) this is necessary. (4) When users are running protocols, especially when they are transmitting messages among nodes, users should prevent other people from interfering this procedure.

3.2.3 Networking Properties

Properties The networking properties of the human-controlled channels related to this dissertation are mainly *topological properties*, *channel speed* and *processing capabilities*. They are explained below.

The topological properties are one of the most important networking properties. Firstly, the human-controlled channels can be divided as one-way $\Longrightarrow$ or two-way channels $\Longleftrightarrow$. For example, suppose there is a one-way channel from a slave node S to a coordinator node C, we can represent this channel as $\mathsf{S} \Longrightarrow \mathsf{C}$. If there is a two-way channel between them, we can represent this channel as $\mathsf{S} \Longleftrightarrow \mathsf{C}$.

In addition, the channels can also be divided as one-to-one (one transmitter and one receiver), multiple-to-one (multiple transmitters and one receiver), one-to-multiple (one transmitter and multiple receivers) and multiple-to-multiple channels (multiple transmitters and multiple receivers).

Channel speed is another important networking property. It refers to the data communication speed expressed typically in bits per second (bps). In this dissertation, we only roughly categorize the channels as high-speed and low-speed channels.

The third is processing capabilities. It refers to how many different messages of particular sizes can be processed by the nodes in a specified time. In this thesis, we use it to distinguish the capabilities of human users and computing devices, i.e. if they can send or receive many different messages in a short time.

Influence to protocol design The networking properties significantly influence the protocol design. For example, when there is a high-speed OOB channel, we can transfer long

strings, e.g. long hash output, via this channel directly; but if the channel is a low-speed one, a short string might be a better choice. The second example is related to topological properties and processing capabilities: it is generally difficult to ask a human user to compare many LEDs at the same time; but if the receiver is a camera, it can still be of an elegant design. Therefore, when we design protocols, the networking properties of OOB channels must be considered.

3.3 HISPs in Personal Networks

This section includes: (1) human interactive security protocols; (2) efficiency improvement using a keyed digest function; (3) security improvement using a group commitment scheme; and (4) example protocols.

3.3.1 Human Interactive Security Protocols

Overview In human interactive security protocols, we usually use NSB human-controlled channels in order to compare strings/digits/a series of patterns and input confirmation signals. We can verify the data transmitted via wireless radio channels; and we can thus bootstrap security in personal networks. A general definition is given as follows.

Definition 3.3 (Human interactive security protocol (HISP)). A HISP is a protocol that bootstraps security in personal networks based on the security properties of human-controlled OOB channels.

This interesting type of protocol is studied in both two-party and group communication scenarios by Balfanz et al. [14], Creese et al. [31], Gehrmann et al [43], Vaudenay [109], Cagalj et al. [20], Wong and Stajano [119], and Roscoe and Nguyen[81, 82]. More human interactive security protocols can be found in the survey paper [82].

Examples With HISPs, we can easily keep entity authenticity, data authenticity and integrity in personal networks. Two examples are given below.

Example (HISP I). This example shows that the NSB human-node channels are helpful when sensor data are transmitted over insecure data communication channels. The threat

Protocol 3.1 (Example HISP I)**.**

1. $\mathsf{S} \longrightarrow \mathsf{C} : T$
2. $\mathsf{S} \Longrightarrow_{display}$ Patient : T
3. $\mathsf{C} \Longrightarrow_{display}$ Patient : T'

model and assumptions are

- $\Longrightarrow_{display}$ is a NSB HCC.
- S, C and Patient are trustworthy.

The message flow of the example protocol is shown as Protocol 3.1, and is explained as follows.

1. S sends temperature data T to C.
2. S shows the data to the patient using a small display:
3. C shows received temperature data T' on its display, which will be checked by the patient. The patient compares T and T'; if $T = T'$, the transmission is correct. □

Protocol 3.2 (Example HISP II)**.**

4. $\mathsf{C} \longrightarrow DDoc : T'$
5. $DDoc \Longrightarrow_{display}$ Doctor : T''
6. Doctor $\Longrightarrow$ Patient : T''

Example (HISP II). This example, which is a continuation of Example HISP I, shows that the NSB human-node and human-human channels are helpful when sensor data are transmitted over insecure data communication channels. The threat model and assumptions are

- $\Longrightarrow_{display}$ and $\Longrightarrow$ are NSB HCCs.
- S, C, Doctor and Patient are trustworthy.

The message flow of the example protocol is shown as Protocol 3.2, and is explained as follows.

4. C sends T' to doctor's device $DDoc$.

5. $DDoc$ shows received temperature data T'' on its display.

6. The patient and doctor compare T' and T''; if $T' = T''$, the transmissions are correct.

□

3.3.2 Efficiency Improvement Using Keyed Digest Function

Efficiency is always important to HISPs. First of all, users are required to compare strings/digits/pattern series, but human users find it difficult and unpleasant to compare long strings/digits/pattern series. Moreover, the human-node channels are always established using small displays, even LEDs and buzzers, which are unable to convey a large amount of information efficiently. Thus, in order to use human-controlled channels in personal networks, the strings/digits/pattern series must be short.

Keyed digest function and cryptographic hash function A frequently used form of short strings is a digest; it is always helpful to balance the security and efficiency (user burden) [81, 82]. The definition of a keyed digest function is as follows.

Definition 3.4 (Keyed digest function). A keyed digest function *digest* takes as input a message msg of arbitrary length and produces as output a short digest value D of fixed length b. In other words, If $\mathbb{K}$ is a key domain, $key \in \mathbb{K}$, the keyed digest function $digest(key, msg) = D$, where the length b of the output D is usually 16 or 32 bits. In addition, *digest* should have the following properties:

- For any msg, $digest(key, msg)$ is uniformly distributed as key varies.
- For any two messages $msg_1 \neq msg_2$, as key varies, the probability

$$\Pr[digest(key, msg_1) = digest(key, msg_2)] \leq \epsilon,$$

  where ϵ is a very small number and typically ϵ is 2^{-b} or slightly more.

It is necessary to distinguish the digest function from a cryptographic hash function. The hash function *hash* has a similar definition as the digest function except that the output

is much longer. Today, the output of a secure hash function is generally longer than 160 bits.

Definition 3.5 (Cryptographic hash function). A cryptographic hash function $hash$ takes as input a message msg of arbitrary length and produces as output a hash value H of fixed length. In other words, $hash(msg) = H$. In addition, $hash$ should have the following properties:

- It is easy to compute $hash(msg) = H$ for any msg.
- Preimage resistance: given H, it is difficult[1] to find msg such that $H = hash(msg)$.
- Second-preimage resistance: given msg_1, it should be difficult to find another input msg_2, where $msg_1 \neq msg_2$, such that $hash(msg_1) = hash(msg_2)$.
- Collision resistance: it should be difficult to find two messages msg_1 and msg_2, where $msg_1 \neq msg_2$, such that $hash(msg_1) = hash(msg_2)$.

Combinatorial attack The short digest string D is easy to be shown, and easy to be compared. However, the digest function does not have second-preimage resistance and collision resistance. Adversary A can compromise the digest by a combinatorial search; we name this attack as a *combinatorial attack*.

Definition 3.6 (Combinatorial attacks). Given msg_1, msg_2 and key, the combinatorial attacks include the following circumstances:

- A can find key_1 such that $digest(key_1, msg_1) = digest(key_1, msg_2)$;
- A can find key_2 such that $digest(key_2, msg_1) = digest(key, msg_2)$;
- A can find more quickly key_3 and key_4, such that

$$digest(key_3, msg_1) = digest(key_4, msg_2).$$

[1] finding a digest collision becomes a matter or pure luck which cannot be improved by any feasible analysis. Clearly the larger the range of digest functions, and the more uniformly distributed they are, the more luck is required.

Example The example below shows one example protocol using the digest function and the combinatorial attack. This protocol exchanges identities between S and C; and S submits accumulated very long sensor data SD_{S} to C. The sensor information is long, thus direct comparisons using HCCs, for example using Example HISP I and II, is not efficient. Thanks to the digest function, users need only compare 16 or 32 bits short digests. In addition, the digest output is more easily shown by nodes.

Protocol 3.3 (Flawed HISP III using digest)**.**

1. $\mathsf{C} \longrightarrow \mathsf{S} : \mathsf{C}$
2. $\mathsf{S} \longrightarrow \mathsf{C} : \mathsf{S}, SD_{\mathsf{S}}$
3. $\mathsf{C} \Longrightarrow_{display}$ User : D_{C}
4. $\mathsf{S} \Longrightarrow_{display}$ User : D_{S}

Example (Flawed HISP III using digest). Suppose SD_{S} is the accumulated sensor data of S; and SD'_{S} is the sensor data that C has received from S. The threat model and assumptions are

- $\Longrightarrow_{display}$ is a NSB HCC.
- The user and all the nodes are trustworthy.
- Button pushing procedure is a NSB HCC.

The message flow is shown in Protocol 3.3, and is explained as follows.

1. C sends S its identity C over a wireless radio channel.
2. S sends C its identity S and accumulated long sensor data SD_{S} over a wireless radio channel.
3. C computes $D_{\mathsf{C}} = digest(\mathsf{C}, \mathsf{S}, SD'_{\mathsf{S}})$. C shows D_{C} using its display.
4. S computes $D_{\mathsf{S}} = digest(\mathsf{C}, \mathsf{S}, SD_{\mathsf{S}})$. S shows D_{S} using its small display. If $D_{\mathsf{C}} = D_{\mathsf{S}}$, user pushes the confirmation buttons on C and S. □

However, a combinatorial attack is possible. Since the digest is vulnerable to combinatorial attacks, attacker A can search $SD_{\mathsf{A(S)}}$ that fulfils

$$digest(\mathsf{C}, \mathsf{S}, SD_{\mathsf{A(S)}}) = digest(\mathsf{C}, \mathsf{S}, SD_{\mathsf{S}}).$$

In the end, C and S will show the same short digest strings, but the protocol is compromised.

3.3.3 Security Improvement Using the Group Commitment Scheme

In order to eliminate the combinatorial attacks mentioned in the last subsection, we introduce the commitment scheme. In addition, in order to eliminate a group size attack, we extend the commitment scheme to a group version.

Commitment scheme A commitment scheme [110] includes two phases:

- Commit phase. A secret value Θ is chosen and sent out by a party C. The message that contains Θ is named as the commitment. Θ can only be known by C during this phase; and this is named as the hiding property.
- Reveal phase. Θ is revealed and checked. The message that reveals Θ is named as the opening. Θ must be the only one that can be computed and that can be validated during the reveal phase; and this is named as the binding property.

One useful commitment scheme is *hash commitment*, which uses $hash(\Theta)$ as the commitment. Hash commitment can be used to eliminate combinatorial attacks against the digest function in two party HISPs. Protocol 3.4 is one example.

Protocol 3.4 (HISP IV using hash commitment)**.**

1. $\mathsf{C} \longrightarrow \mathsf{S} : \mathsf{C}, hash(N_{\mathsf{C}}, \mathsf{C})$
2. $\forall \mathsf{S} \longrightarrow \mathsf{C} : \mathsf{S}, SD_{\mathsf{S}}$
3. $\mathsf{C} \longrightarrow \forall \mathsf{S} : N_{\mathsf{C}}$
4. $\mathsf{C} \wedge \mathsf{S} \Longrightarrow_{display}$ User $: D$

Example (HISP IV using hash commitment). Suppose C and sensor S want to exchange identities, and S wants to submit the accumulated very long sensor data SD_{S} to C. We can

design a protocol using hash commitment and digest. The threat model and assumptions are

- $\Longrightarrow_{display}$ is a NSB HCC.
- The user and all the nodes are trustworthy.
- The notification/verification signal (e.g. LED alarm signal that should be read by human users) is transmitted over a NSB HCC.
- Button pushing procedure is a NSB HCC.

The message flow is shown in Protocol 3.4; it is explained as follows.

1. C sends S its identity C and hash commitment $hash(N_{\mathsf{C}}, \mathsf{C})$ over a wireless radio channel. N_{C} is a long random string.
2. After S has received message 1, it sends C its identity S and sensor data SD_{S} over a wireless radio channel. *After it sends out message 2, it flashes its LED.*
3. If the user sees that S has flashed its LED, the user pushes a button on C to release message 3: the opening N_{C}. S verifies $hash(N_{\mathsf{C}}, \mathsf{C})$.
4. C and S generate and show $D = digest(N_{\mathsf{C}}, \mathsf{C}, \mathsf{S}, SD_{\mathsf{S}})$ on displays. The user compares D shown by each device. If they are the same, the user pushes a button on each device. □

The combinatorial attack against the digest is eliminated. Actions in steps 2 and 3 separate the commit phase (step 1 and step 2) and reveal phase (step 3 and step 4). The attacker does not know the digest value in the commit phase. Thus, the attacker cannot initiate a combinatorial search, which means that the chance of finding fake identities and sensor data such that the same digests will be generated in the final step is negligible.

Group commitment scheme In a group scenario, i.e. one coordinator node and several slave nodes, the commitment scheme should be extended. The reason is that the commitment scheme does not have a group-size control mechanism, thus attack is still possible.

The example below shows a group-size attack against a flawed group protocol extended from Protocol 3.4.

Attack 3.1 (Group-size attack)**.**

1. $\mathsf{C} \longrightarrow \mathsf{S} \wedge \mathsf{A} : \mathsf{C}, hash(N_{\mathsf{C}}, \mathsf{C})$
2. $\mathsf{A} \longrightarrow \mathsf{C} \wedge \mathsf{S} : \mathsf{A}, SD_{\mathsf{A}}$
3. $\mathsf{S} \longrightarrow \mathsf{C} \wedge \mathsf{A} : \mathsf{S}, SD_{\mathsf{S}}$
4. $\mathsf{C} \longrightarrow \mathsf{S} \wedge \mathsf{A} : N_{\mathsf{C}}$
5. $\mathsf{C} \wedge \mathsf{S} \Longrightarrow_{display} \text{User} : D$

Example (Group-size attack). Suppose there are two members C and S; and there is one intruder A. The message flow of the group-size attack is shown in Attack 3.1. The steps are explained as follows.

1. C sends S its identity C and hash commitment $hash(N_{\mathsf{C}}, \mathsf{C})$ over wireless radio channels. N_{C} is a long random string. Meanwhile, attacker A eavesdrops this message.

2. Attacker A sends out its identity A and fake sensor data SD_{A} to C and S.

3. After S has received message 1, it sends C its identity S and sensor data SD_{S} over wireless radio channels. After this, it flashes its LED.

4. After the user sees that S has flashed its LED, the user pushes a button on C to release message 4: the opening N_{C}.

5. C and S generate and show $D = digest(N_{\mathsf{C}}, \mathsf{C}, \mathsf{S}, SD_{\mathsf{S}}, \mathsf{A}, SD_{\mathsf{A}})$ on displays. The user compares the D shown by each device. They are the same; the protocol *fails to find out that the intruder named A has joined in the group.* □

In order to prevent combinatorial attacks and group-size attacks, we extend the commitment scheme to a group version. One round of the group commitment protocol allows members to commit to a secret value Θ; it also allows this value to be revealed and checked later. The beginning point is named *bp*, and the end point is named *ep*. The related concepts are defined below.

$$\frac{\text{One round}}{bp \rightarrow cp \rightarrow rp \rightarrow ep}$$

Figure 3.4: Temporal model of group commitment protocol

Definition 3.7 (Commit phase and commit point). In the commit phase, all members exchange partial commitment messages (all or part of the messages are selected as the construction information of Θ). Using these messages, members cooperatively choose a secret value Θ. The time that all the group members cease to accept new commitment messages is named as commit point *cp*. Note that a *group-size check* should always be explicitly or implicitly included in this phase before *cp*.

Definition 3.8 (z-knowing property). If there are only z members ($gs > z \geq 0$, where gs is the group size) who know the value Θ legitimately during the commit phase, this protocol fulfils z-knowing property.

Definition 3.9 (Reveal phase and reveal point). In the reveal phase, all members exchange partial opening messages (revealing how Θ is constructed). Using these messages, Θ is revealed and checked. The time that a node (might be A) that is not one of the z-knowing members knows the value of Θ is named as reveal point *rp*.

Definition 3.10 (Binding property). During the reveal phase, Θ must be the only value that all the members can compute and that validates. This is called the binding property.

Definition 3.11 (Group commitment scheme). A group commitment scheme consists of two phases: the commit phase (group-sized check explicitly or implicitly included) and the reveal phase. In addition, the protocol holds the z-knowing and binding property.

- If $z = 0$, the group commitment protocol is a group commitment agreement protocol.
- If $z = 1$, the group commitment protocol is a group commitment transport protocol.
- If $z > 1$, the group commitment protocol is a hybrid group commitment protocol.

Fig. 3.4 is a temporal model of the group commitment protocol. Note that the scheme requires that the commit phase and the reveal phase are separated. The time point *cp* is critical in this, and needs to be carefully designed.

The group commitment scheme can be used to eliminate combinatorial attacks against HISPs. The main idea is that all the legal parties exchange only the useful information in the commit phase. Suppose A wants to change the messages in the commit phase. Because A does not know the final digest value in this phase, the chance of finding proper fake messages that do not influence the final digest value is negligible. Thus, combinatorial attacks are eliminated.

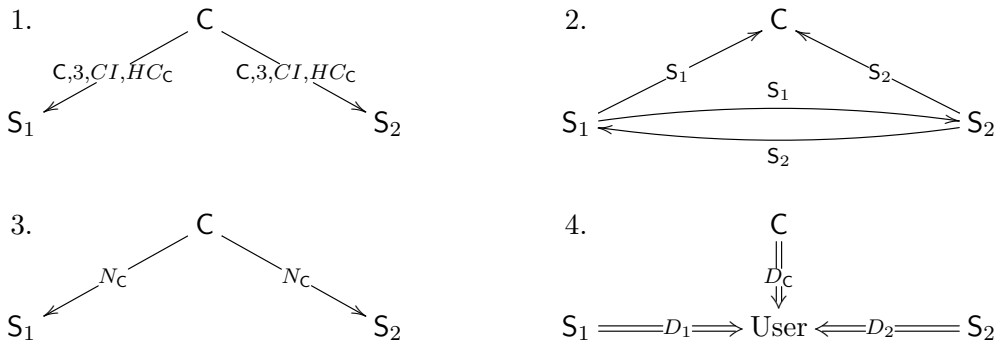
Group size attack is also eliminated. In the commitment phase, group size is checked and committed. Note that the user knows the group size in our scenario. Thus, the chance of passing the group size check is negligible (break the commitment scheme).

In the next subsection, we will provide two example HISPs with group commitment scheme: HISP-CI[2] and HISP-SD. HCBK and SHCBK, which are general versions of HISP-CI and HISP-SD, have been *formally checked using FDR*, and no attacks were found; related information can be found in [27, 81, 82].

3.3.4 Example: HISP-CI

HISP-CI is one example of HISP with the group commitment scheme. Suppose there is a personal network that consists of one mobile phone C, and two slave nodes S_1 and S_2. HISP-CI helps C to deliver its instructions CI securely. The message flow is shown in Protocol. 3.5. The details are explained below.

Protocol 3.5 (HISP-CI)**.**



Commit phase step 1. (a) *bp*: the user pushes a buttons on these nodes in order to

[2]C for coordinator, I for instruction

initiate the protocol. (b) The user inputs the group size gs on C. (c) After gs has been received, C generates a long random number N_{C}. C broadcasts its identity C, gs, instructions CI, and the commitment $HC_{\mathsf{C}} = hash(N_{\mathsf{C}}, \mathsf{C}, gs)$ over wireless radio channels.

Commit phase step 2. (a) After message 1 has been received, the two slave nodes broadcast their identity S_1 and S_2 over wireless radio channels. (b) Each slave node counts the number of received different commitment messages. If the number is $(3-1)$, it ceases to accept any new commitment message, and then it flashes LEDs. (c) C counts the number of received different commitment messages. If the number is $(3-1)$, it ceases to accept any new commitment message, and flashes its LEDs. (d) *cp*: when all the nodes have flashed their LEDs, the user pushes a button on C to release message 3.

Reveal phase step 3. (a) *rp*: C broadcasts the opening message N_{C} over wireless radio channels. (b) Each slave node verifies HC_{C}. If the verification fails, this slave node aborts and informs the user.

Reveal phase step 4. (a) Each node computes and shows the digest value on its display. (b) The user verifies that all the digests are the same:

$$D_{\mathsf{C}} = D_1 = D_2 = digest(N_{\mathsf{C}}, \mathsf{C}, \mathsf{S}_1, \mathsf{S}_2, CI).$$

(c) *ep*: if the verification succeeds, the user pushes a button on each node.

Remark. The threat model and assumptions are

- HCCs using displays are NSB channels.
- The user and all the nodes are trustworthy.
- group size is inputted via a NSB HCC.
- The notification/verification signal (e.g. LED alarm signal that should be read by human users) is transmitted over a NSB HCC.
- Button pushing procedure is a NSB HCC.

This protocol uses displays, and thus it is appropriate to devices with displays, e.g. tablets, mobile phones, and smart watches.

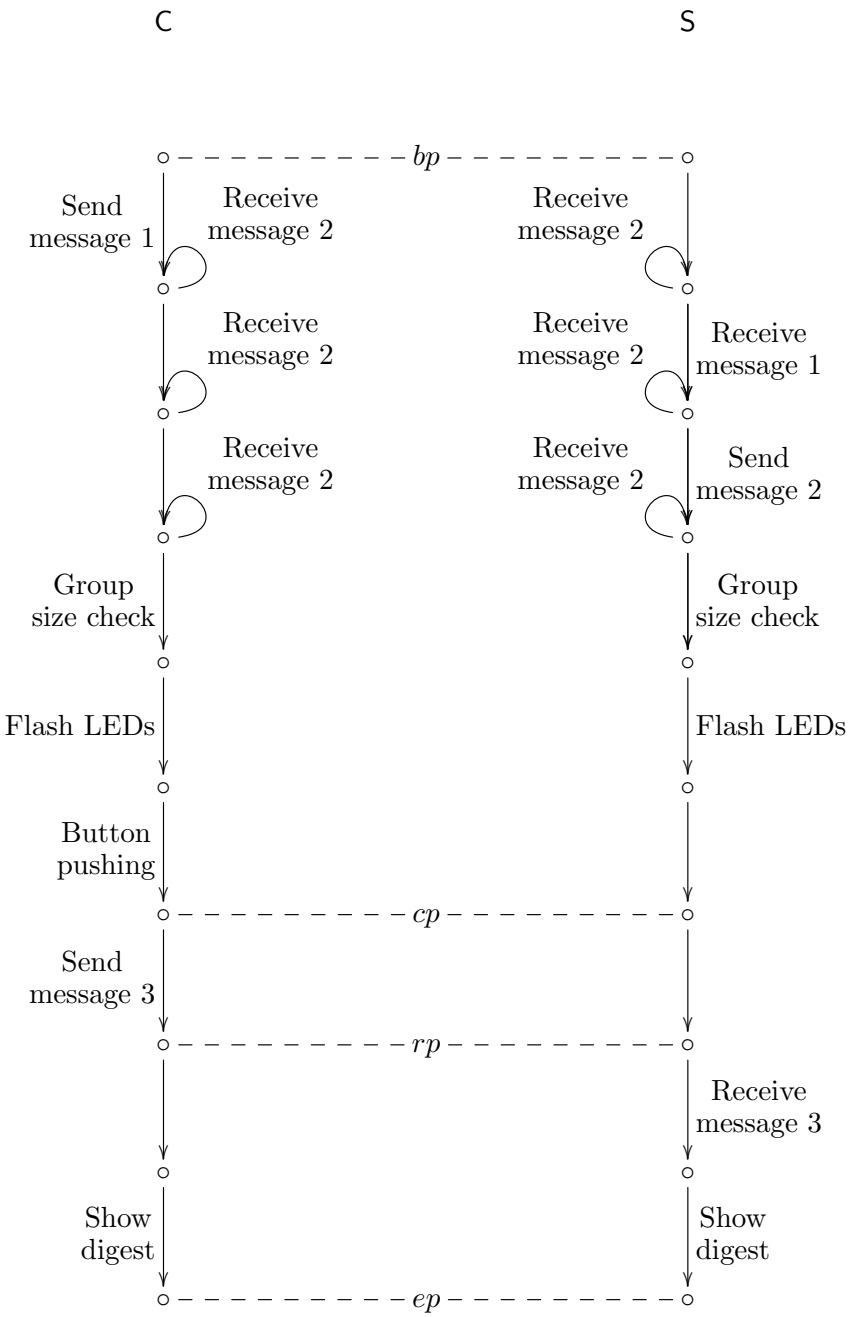


Figure 3.5: One round of HISP-CI.

Understanding HISP-CI We mainly focus on the question: have C, S_1 and S_2 agreed on $info = \{\mathsf{C}, \mathsf{S}_1, \mathsf{S}_2, CI\}$?

Before the analysis, a clear picture of a protocol run is helpful. A protocol run is shown in Fig. 3.5 (The procedure of C is shown on the left, and the procedure of S is shown on the right). Time point *bp* corresponds to the button pushing in step 1.a; *cp* corresponds to

the button pushing in step 2.d; *rp* corresponds to step 3.a; and *ep* corresponds to step 4.c. We now show that C has successfully finished one protocol run. The analysis is below.

The probability of the event: A knows nonce N_{C} before *cp* is bounded by the probability of compromising the hiding property or ϵ (mentioned in the definition of the digest function). In order to know N_{C} before *cp*, A has two strategies:

- A breaks the hiding property. It is negligible.
- A generates a fake nonce by himself. However, A must generate the fake nonce without knowing the final digest value before *cp*. In this case, the probability that A will succeed is bounded by ϵ.

Thus, we can neglect the chance that A knows N_{C} before *cp*. However, is it possible that A compromise the protocol by changing message 3? This probability is bounded by the probability of compromising the binding property or ϵ. There are only two cases.

- HC_{C} is not changed. A is required to compromise the binding property.
- HC_{C} is also changed. In other words, A generates a fake nonce before *cp*. As we analyzed previously, the probability that A succeeds is bounded by ϵ.

Thus, we can neglect the chance that A (1) knows N_{C} before *cp*, and (2) changes message 3. However, is it possible that A compromises the protocol by changing message 1 or 2? This probability is bounded by the probability of compromising the binding property or ϵ. There are only two cases.

- HC_{C} is also changed. The binding property is compromised.
- HC_{C} is not changed. If $info$ is changed before *cp*, the probability that A succeeds is bounded by ϵ.

Therefore, when C has successfully finished one protocol run, the chance that $Info$ has been changed is negligible. Similarly, when S_1 or S_2 has successfully finished one protocol run, the chance that $Info$ has been changed is negligible. In addition, since we use NSB HCCs in step 4, all the nodes have been involved in the same protocol run. Thus, all the nodes have agreed on $Info$.

Understanding group commitment scheme in HISP-CI We will now look at the second question: is it possible that A initiates group size attacks? Firstly, C gets gs via a NSB HCC, thus A cannot modify gs on C.

Secondly, S_1 and S_2 get gs from message 1. Suppose gs has been maliciously changed. There are only three cases.

- Either HC_{C} or message 3 is changed. The binding property is compromised.
- Both HC_{C} and message 3 are changed. This is bounded by ϵ.
- Neither HC_{C} nor message 3 is changed. The cryptography hash function is compromised.

Thus, it is negligible that A can maliciously change gs on S_1 or S_2.

We understand the group commitment scheme as follows. The commit phase is step 1 and step 2. $HC_{\mathsf{C}} = hash(N_{\mathsf{C}}, \mathsf{C}, gs)$ is the group commitment message. It fulfils the 1-knowing property: only C knows the secret value N_{C} in the commit phase. In addition, authentic group size is checked in each node before *cp*.

The reveal phase is step 3 and step 4. In these steps, N_{C} is revealed. Thanks to the properties of cryptographic hash functions, N_{C} is the only value that can be computed and that can be validated. Thus, it fulfils the binding property.

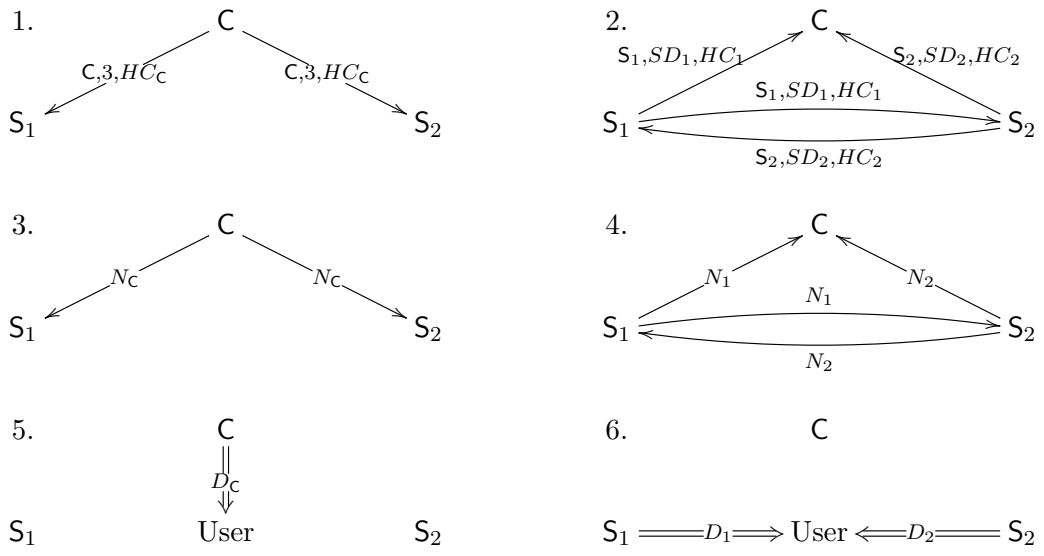
Also, since N_{C} is hidden in the commit phase, the final digest value is also a secret in this phase; thus, A cannot initiate combinatorial attacks. However, if we understand group commitment scheme in this way, the combinatorial attack elimination is only a "side effect". In the next protocol, we will try to understand the group commitment scheme in a different way.

3.3.5 Example: HISP-SD

HISP-SD is another example of HISP that helps two slave nodes S_1 and S_2 to submit their sensor data SD_1 and SD_2 securely. The message flow is shown in Protocol 3.6. The details are explained below.

Commit phase step 1. (a) *bp*: the user pushes buttons on the nodes in order to initiate the protocol. (b) The user inputs group size gs on C. (c) C generates a long random number N_{C}.

Protocol 3.6 (HISP-SD)**.**



(d) C broadcasts its identity C, group size gs, and hash commitment $HC_C = hash(N_C, C)$ over wireless radio channels.

Commit phase step 2. (a) After message 1 has been received, each slave node generates a long random number (N_1, N_2). Each slave node broadcasts its identity (S_1, S_2), sensor data (SD_1, SD_2), and hash commitment ($HC_1 = hash(N_1, S_1)$, $HC_2 = hash(N_2, S_2)$) over wireless radio channels. (b) *cp*: C counts the number of received different commitment messages. If the number is $(3-1)$, it ceases to accept any new commitment message, and goes to step 3. (c) *cp*: each slave node counts the number of received different commitment messages. If the number is $(3-1)$, it ceases to accept any new commitment message, and goes to step 3; if it receives messages 3 before $(3-1)$ different commitment messages has been received, it aborts and informs the user.

Reveal phase step 3. (a) *rp*: C sends the opening message N_C to each slave node over wireless radio channels. (b) Each slave node verifies HC_C. If the verification fails, the slave node aborts and informs the user.

Reveal phase step 4. (a) If step 3.b succeeds, each slave node broadcasts its opening message (N_1, N_2) over wireless radio channels. (b) Other nodes verify HC_1 and/or HC_2. If the verification fails, the node aborts and informs the user.

Reveal phase step 5. C computes and shows D_{C} on its display.

$$D_{\mathsf{C}} = digest(N_{\mathsf{C}} \oplus N_1 \oplus N_2, 3, \mathsf{C}, \mathsf{S}_1, SD_1, \mathsf{S}_2, SD_2)$$

Reveal phase step 6. (a) Each slave node computes and shows the digest (D_1 and D_2) on its display.

$$D_1/D_2 = digest(N_{\mathsf{C}} \oplus N_1 \oplus N_2, 3, \mathsf{C}, \mathsf{S}_1, SD_1, \mathsf{S}_2, SD_2)$$

(b) *ep*: the user verifies $D_1 = D_2 = D_{\mathsf{C}}$. If verification succeeds, the user pushes a button on each node.

Remark. The threat model and assumptions are

- HCCs using displays are NSB channels
- The user and all the nodes are trustworthy.
- group size is inputted via a NSB HCC.
- The notification/verification signal (e.g. LED alarm signal that should be read by human users) is transmitted over a NSB HCC.
- Button pushing procedure is a NSB HCC.

This protocol uses displays, and thus it is appropriate to devices with displays, e.g. tablets, mobile phones, and smart watches.

Understanding HISP-SD We mainly focus on the question: have C, S_1 and S_2 agreed on $info = \{\mathsf{C}, \mathsf{S}_1, SD_1, \mathsf{S}_2, SD_2\}$?

Before the analysis, a clear picture of one protocol run is helpful. One protocol run is shown in Fig. 3.6 (The procedure of C is shown on the left, and the procedure of S is shown on the right). We now show that C has successfully finished one protocol run. The analysis is below.

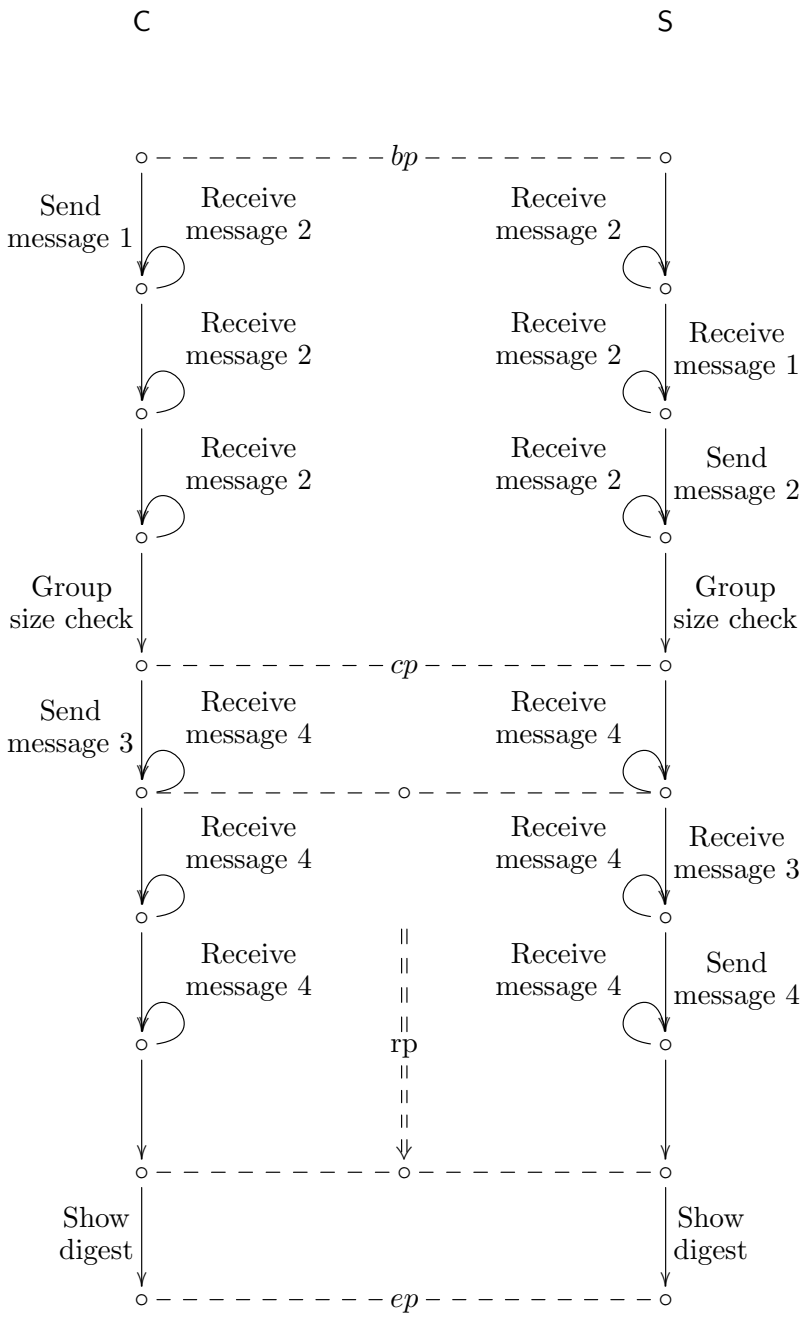


Figure 3.6: One round of HISP-SD.

The probability of the event: A knows nonce N_C before *cp*, is bounded by the probability of compromising the hiding property or ϵ. In order to know N_C before *cp*, A has two strategies:

- A breaks the hiding property.

- A generates a fake nonce by himself. However, A must generate the fake nonce without knowing the final digest value before *cp*. In this case, the probability that A will succeed is bounded by ϵ.

Thus, we can neglect the chance that A knows N_{C} before *cp*. However, is it possible that A compromises the protocol by changing message 3 or 4? This probability is bounded by the probability of compromising the binding property or ϵ. There are only two cases.

- HC_{C}, HC_1 and HC_2 are not changed. A is required to compromise the binding property.

- HC_{C} , HC_1 or HC_2 is also changed. In other words, A generates at least one fake nonce in the commit phase. This probability is bounded by ϵ.

Thus, we can neglect the chance that A (1) knows N_{C} before *cp*, and (2) changes message 3 or 4. However, is it possible that A compromises the protocol by changing message 1 or 2? This probability is bounded by the probability of compromising the binding property or ϵ. There are only two cases:

- HC_{C}, HC_1 or HC_2 is changed. The binding property is compromised.

- HC_{C}, HC_1 and HC_2 are not changed. In order to change $info$, the probability that A will succeed is bounded by ϵ.

Therefore, when C has successfully finished one protocol run, the chance that $Info$ has been changed is negligible. Similarly, when S_1 or S_2 has successfully finished one protocol run, the chance that $Info$ has been changed is negligible. In addition, since we use NSB HCCs in step 4, all the nodes have been involved in the same protocol run. Thus, all the nodes have agreed on $Info$.

Understanding group commitment scheme in HISP-SD We will now look at the second question: is it possible that A initiates group size attacks? Firstly, C gets gs via a NSB HCC; thus, A cannot modify gs on C.

Secondly, S_1 and S_2 get gs from message 1. Suppose gs has been maliciously changed. There are only three cases.

- (a) HC_{C} , HC_1 or HC_2 is changed, but message 3 or 4 is not. (b) HC_{C} , HC_1 and HC_2 are not changed, but message 3 or 4 is. The binding property is compromised.
- Message 3 or 4 is changed, and HC_{C}, HC_1 or HC_2 is also changed correspondingly. The probability that A will succeed is bounded by ϵ.
- None of HC_{C}, HC_1, HC_2, message 3, and message 4 is changed. The probability that A will succeed is bounded by ϵ.

Thus, we can neglect the chance that A can maliciously change gs on S_1 or S_2.

We understand the group commitment scheme in HISP-SD as follows. The commit phase is step 1 and step 2. Message 1 and message 2 are group commitment messages. This protocol fulfils the 0-knowing property: none of the nodes knows the final digest value, which is the secret chosen in the commit phase. In addition, the authentic group size is checked in each node before *cp*.

The reveal phase is step 3 and step 4. In these steps, nonces are revealed; and thus the secret digest value is revealed. This is the only value that can be computed and that can be validated. Thus, it fulfils the binding property.

Note that we regard the digest value as the secret value Θ here; however, in HISP-CI, the nonce is Θ. Either one of these understanding methods can explain why the combinatorial attacks and group size attacks are eliminated.

3.4 Chapter Summary

In this chapter, we have studied human-controlled channels and HISPs in personal networks. They can protect the messages transmitted over insecure wireless radio channels.

From the chapter, we can see that many human-controlled channels are available. Generally speaking, they are low-speed NSB OOB channels. Their security and networking properties will influence the design of HISPs.

In order to improve efficiency and security, the keyed digest function and the group commitment scheme are used. The keyed digest function significantly increases efficiency.

In addition, the combinatorial attacks and group size attacks are eliminated by the group commitment scheme.

HISPs are flexible. They *do not need pre-distributed secrets* to bootstrap security in wearable personal networks. Whenever the existing security credentials are compromised, users can *easily update* them using HISPs.

Chapter 4

Use Cases: Human Interactive Security Protocols

In this chapter, we will study two HISP[1] use cases. These use cases will show that HISPs are useful building blocks of security mechanisms in wearable personal networks.

The first use case is the Elliptic Curve Diffie-Hellman (ECDH) version of HISPs. These protocols provide a possible way of allocating identities and keys in personal networks. Thanks to them, the man-in-the-middle attack, which is the main problem of ECDH, is eliminated. In addition, compromised and expired keys can be easily changed.

In addition, binding protocols are designed as our second use case. These protocols can eliminate Sybil attacks. In some scenarios, these protocols can be used instead of key distribution protocols.

Contents

[1] Human interactive security protocol

4.1 ECDH Version of HISPs

In this section, we will introduce the first use case: ECDH version of HISPs. This use case is organized as follows: (1) ECDH protocol used in [70], and man-in-the-middle (MITM) attacks against it; (2) HISP-K[2] I; (3) HISP-K II; and (4) discussion.

4.1.1 ECDH and Attacks

ECDH overview ECDH is currently used for key distributions in sensor networks. For example, it is implemented in TinyECC [70], which is one of the most popular public key cryptography libraries for wireless sensor networks. Since ECDH has a smaller key size than for example the RSA-based Diffie-Hellman protocol, it is a good solution for the memory-limited nodes.

ECDH allows two nodes, e.g., C and S, to establish a shared secret over an insecure channel. The fundamental theory behind this is as follows: it is hard to solve the elliptic curve discrete logarithm problem; thus it is believed that finding private token EV from corresponding public token EU with the knowledge of the base point G is infeasible. Thus, C and S can exchange their public tokens over insecure channel without leaking the private token. After C and S exchange their public tokens, each one can compute a shared secret using its private token and its opponent's public token.

Example protocol steps Before running ECDH, nodes should prepare the following parameters.

- Coordinator and slave nodes agree on the elliptic curve, the cofactor, the public base point G and its order;
- Coordinator node generates a random integer EV_{C} as its private token, and $EU_{\mathsf{C}} = [EV_{\mathsf{C}}]\mathsf{G}$ as its public token; slave node generates a random integer EV_i as its private token, and $EU_i = [EV_i]\mathsf{G}$ as its public token.

The ECDH message flow is shown in Protocol 4.1. The details are explained below.

[2] K for key establishment protocol

Protocol 4.1 (ECDH)**.**

1. $\mathsf{C} \longrightarrow \mathsf{S}_i : EU_{\mathsf{C}}$
2. $\mathsf{S}_i \longrightarrow \mathsf{C} : EU_i$

Step 1. C sends S_i *message 1* including its public token EU_{C}. S_i computes shared ECDH key $K_i = [EV_i]EU_{\mathsf{C}}$.

Step 2. S_i sends C *message 2* including its public token EU_i. C computes shared ECDH key $K_i = [EV_{\mathsf{C}}]EU_i$.

MITM attack It is well known that ECDH is vulnerable to the MITM attack; one attack instance is shown as Attack 4.1. The attack includes four steps explained below.

Attack 4.1 (ECDH MITM Attack)**.**

1. $\mathsf{C} \longrightarrow \mathsf{A}(\mathsf{S}_i) : EU_{\mathsf{C}}$
2. $\mathsf{A}(\mathsf{S}_i) \longrightarrow \mathsf{C} : EU_{\mathsf{A}}$
3. $\mathsf{A}(\mathsf{C}) \longrightarrow \mathsf{S}_i : EU_{\mathsf{A}}$
4. $\mathsf{S}_i \longrightarrow \mathsf{A}(\mathsf{C}) : EU_i$

Step 1. C sends attacker A, who pretends to be S_i, *message 1*: its public token EU_{C}. A computes shared ECDH key $K_i = [EV_{\mathsf{A}}]EU_{\mathsf{C}}$.

Step 2. Attacker A, who pretends to be S_i, sends C *message 2*: its public token EU_{A}. C computes shared ECDH key $K_i = [EV_{\mathsf{C}}]EU_{\mathsf{A}}$.

Step 3. Attacker A, who pretends to be C, sends S_i *message 3*: its public token EU_{A}. S_i computes shared ECDH key $K'_i = [EV_i]EU_{\mathsf{A}}$.

Step 4. S_i sends attacker A, who pretends to be C, *message 4*: its public token EU_i. A computes shared ECDH key $K'_i = [EV_{\mathsf{A}}]EU_i$.

In this case, C thinks it has a share key K_i with S_i, but actually K_i is shared with the attacker A. Meanwhile, S_i thinks it has a share key K'_i with C, but actually K'_i is shared with the attacker A. Any further messages encrypted using K_i and K'_i can be compromised by the attacker A. Hence, the objective of our protocols in the following subsections is that all parties can receive the correct public tokens without worrying about the MITM attack.

4.1.2 HISP-K I

HISP-K I is a revised ECDH. Similar to ECDH, it can establish shared secrets between nodes over insecure channels. However, this protocol does not suffer from ECDH MITM attacks.

In order to prevent ECDH MITM attacks, we combine ECDH with a HISP. In this revised protocol, C and S_i exchange identities and keys in the commitment phase. Because the attacker A cannot know the committed digest value D, A cannot search suitable fake identities and keys during this phase. In this case, ECDH MITM attacks are eliminated.

Procedure HISP-K I is shown in Protocol 4.2. The details are explained below.

Protocol 4.2 (HISP-K I)**.**

1. $\mathsf{C} \longrightarrow \forall\, \mathsf{S}_i : \mathsf{C}, EU_{\mathsf{C}}, gs, HC_{\mathsf{C}}$
2. $\forall\, \mathsf{S}_i \longrightarrow \mathsf{C} \wedge \forall\, \mathsf{S}_j : \mathsf{S}_i, EU_i$
3. $\mathsf{C} \longrightarrow \forall\, \mathsf{S}_i : N_{\mathsf{C}}$
4. $\forall\, \mathsf{S}_i \Longrightarrow \text{User} : D_i$
5. $\mathsf{C} \Longrightarrow \text{User} : D_{\mathsf{C}}$

Commit phase step 1. (a) *bp*: the user inputs group size gs on C, and pushes a buttons on each node in order to initiate the protocol. (b) C generates a long random number N_{C} and a hash commitment $HC_{\mathsf{C}} = hash(N_{\mathsf{C}}, \mathsf{C})$. (c) C sends S_i *message 1* including its identity C, public token EU_{C}, group size gs, and the commitment HC_{C} over wireless radio channels.

Commit phase step 2. (a) After message 1 has been received, each S_i broadcasts *message 2* including its identity S_i and public token EU_i over wireless radio channels. We assume that $i = 1, \cdots, gs-1$, $j = 1, \cdots, gs-1$, and $i \neq j$. (b) C counts the number of received message 2. If the number is $gs-1$, it ceases to accept new message 2, and flashes its LEDs. (c) Each S_i also counts the number of received commitment messages: message 1 and 2; if the number is $gs-1$, it ceases to accept new commitment messages, and it flashes its LEDs. (d) *cp*: after all the nodes have flashed their LEDs, the user informs C to release message 3.

Reveal phase step 3. (a) *rp*: C sends the opening message *message 3* including N_{C} to

each S_i over wireless radio channels. (b) S_i verifies HC_{C}. If the verification fails, S_i aborts and informs the user.

Reveal phase step 4. (a) S_i computes $D_i = digest(N_{\mathsf{C}}, gs, \mathsf{C}, EU_{\mathsf{C}}, \sum(\mathsf{S}_i, EU_i))$, where $\sum(\mathsf{S}_i, EU_i)$ represents concatenation of (S_i, EU_i). (b) Each S_i transfers *message 4* including its digest value D_i to the user via human-controlled NSB channels.

Reveal phase step 5. (a) C computes $D_{\mathsf{C}} = digest(N_{\mathsf{C}}, gs, \mathsf{C}, EU_{\mathsf{C}}, \sum(\mathsf{S}_i, EU_i))$. (b) C transfers *message 5* including its digest value D_{C} to the user via the human-controlled NSB channel. (b) *ep*: the user verifies the following equality for each S_i: $\forall\, D_i = D_{\mathsf{C}}$. If the verification succeeds, the user pushes the confirmation button on each node.

Remark. The threat model and assumptions are

- HCCs are NSB channels
- The user and all the nodes are trustworthy.
- group size is inputted via a NSB HCC.
- The notification/verification signal (e.g. LED alarm signal that should be read by human users) is transmitted over a NSB HCC.
- Button pushing procedure is a NSB HCC.

This protocol uses human-controlled NSB channels, and thus it is appropriate to devices with displays or other human friendly interfaces. Also, the nodes should have enough resource to run ECC algorithms.

4.1.3 HISP-K II

HISP-K II is another revised ECDH. It can also establish shared secrets between nodes over insecure channels. In addition, this protocol does not suffer from ECDH MITM attacks. It prevents ECDH MITM attacks by combining ECDH with a HISP. It relies on the keyed digest function and group commitment scheme.

There are two differences between HISP-K II and HISP-K I. The first is that HISP-K II is able to reduce the burden of human users compared to HISP-K I. The second difference is that it requires more wireless radio communications than HISP-K I.

Protocol explanation HISP-K II is shown in Protocol 4.3. The details are explained below.

Protocol 4.3 (HISP-K II)**.**

1. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : \mathsf{C}, gs, EU_{\mathsf{C}}, HC_{\mathsf{C}}$
2. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} \wedge \forall \mathsf{S}_j : \mathsf{S}_i, EU_i, HC_i$
3. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : N_{\mathsf{C}}$
4. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} \wedge \forall \mathsf{S}_j : N_i$
5. $\forall \mathsf{S}_i \Longrightarrow \text{User} : D_i$
6. $\mathsf{C} \Longrightarrow \text{User} : D_{\mathsf{C}}$

Commit phase step 1. (a) *bp*: the user inputs group size gs on C; and this user pushes the button on each node in order to initiate the protocol. (b) C generates a long random number N_{C} and a hash commitment $HC_{\mathsf{C}} = hash(N_{\mathsf{C}}, \mathsf{C})$. (c) C sends S_i *message 1* including its identity C, group size gs, public token EU_{C} and hash commitment HC_{C} over wireless radio channels.

Commit phase step 2. (a) After message 1 has been received, each S_i generates a long random number N_i and hash commitment $HC_i = hash(N_i, \mathsf{S}_i)$. (b) Each S_i sends C and S_j *message 2* including its identity S_i, public token EU_i and the commitment HC_i over wireless radio channels. (b) *cp*: C counts the number of received message 2. If the number is $gs - 1$, it ceases to accept any new commitment message, and then it goes to step 3. (c) *cp*: each S_i counts the number of received commitment messages: message 1 and message 2. If the number is $gs - 1$, it ceases to accept any new commitment message, and waits for message 3. If it receives message 3 before it receives $gs - 1$ commitment messages, this slave node aborts and informs the user.

Reveal phase step 3. (a) *rp*: C sends the opening message *message 3* including N_{C} to each S_i over wireless radio channels. (b) S_i verifies HC_{C}. If the verification fails, S_i aborts and informs the user.

Reveal phase step 4. (a) After message 3 has been received, each S_i sends C and S_j the opening message *message 4* including N_i over wireless radio channels. (b) C and S_j verify HC_i. If the verification fails, the node aborts and informs the user.

Reveal phase step 5. (a) S_i computes $D_i = digest(N_{\mathsf{C}} \oplus (\bigoplus N_i), gs, \mathsf{C}, EU_{\mathsf{C}}, \sum(\mathsf{S}_i, EU_i))$. $\bigoplus N_i$ represents the result of XOR of all N_i; $\oplus$ is the XOR operator; and $\sum(\mathsf{S}_i, EU_i)$ represents a concatenation of all (S_i, EU_i). (b) Each S_i transfers *message 5* including its digest value D_i to the user via human-controlled NSB channels.

Reveal phase step 6. (a) C computes $D_{\mathsf{C}} = digest(N_{\mathsf{C}} \oplus (\bigoplus N_i), gs, \mathsf{C}, EU_{\mathsf{C}}, \sum(\mathsf{S}_i, EU_i))$. (b) *ep*: C transfers *message 6* including its digest value D_{C} to the user via human-controlled NSB channels. (c) The user verifies the following equality for each S_i: $\forall\, D_i = D_{\mathsf{C}}$. If the verification succeeds, the user pushes the confirmation button on each node.

The threat model and assumptions are

- HCCs are NSB channels
- The user and all the nodes are trustworthy.
- group size is inputted via a NSB HCC.
- The notification/verification signal (e.g. LED alarm signal that should be read by human users) is transmitted over a NSB HCC.
- Button pushing procedure is a NSB HCC.

This protocol uses human-controlled NSB channels, and thus it is appropriate to devices with displays or other human friendly interfaces. Also, the nodes should have enough resource to run ECC algorithms.

4.1.4 Discussion

HISP-K I and II can eliminate MITM attacks against ECDH. Similar to HISP-SD and HISP-CI in the preceding chapter, the authenticity and integrity of ECDH public tokens can be protected. Therefore, unless the attacker coincidentally has the same public tokens and corresponding private tokens, ECDH MITM attacks are eliminated.

Additionally, HISP-K I and II have not introduced the attacks that were mentioned in the last chapter: group size attacks and combinatorial attacks. These attacks are eliminated by the group commitment scheme: firstly, the group size has been checked, thus group size

attacks are eliminated; secondly, the final digest values are kept hidden from attackers using hash functions and long nonces, thus combinatorial attacks are eliminated.

HISP-K I and II are pair-wise key agreement protocols. All the parties influence the generation of ECDH keys. In addition, each pair of nodes share a ECDH key after the protocol run.

4.2 Identity Binding

In this section, we will introduce the second use case of HISPs: identity binding protocols. This use case is organized as follows: (1) identity system and sybil attack, (2) binding protocols, and (3) related discussion.

4.2.1 Identity System and Sybil Attack

Identity systems Identity (ID) systems of personal networks mainly focus on the relationship between nodes and identities. An *identity* can be a *name*, which is a title for denoting things; or it can be an *address*, which supplies the information needed to find these things. For example, unique node identifier is a name, and media access control address is an address. However, this distinction is not a sharp one: an IP address contains information to both find and identify a node. Binding is a mapping between nodes and identities, more specifically, between names and addresses. [59]

Sybil attack review In a sybil attack [79], a single malicious node illegitimately uses multiple identities. This makes other nodes believe in a wrong topology, which can significantly interrupt to media access control layer and routing layer protocols. For example, in Fig. 4.1, attacker A use multiple identities: A_1, A_2, and A_3. In this case, A can compromise the aggregation protocols. Suppose the aggregation result *result* is the average of measurements from all nodes; and the collected measurements are $data_{\mathsf{A}_1}$, $data_{\mathsf{A}_2}$, $data_{\mathsf{A}_3}$, and $data_{\mathsf{S}}$. The aggregation result is as follows:

$$result = (data_{\mathsf{A}_1} + data_{\mathsf{A}_2} + data_{\mathsf{A}_3} + data_{\mathsf{S}}).$$

Therefore, A can easily change *result*.

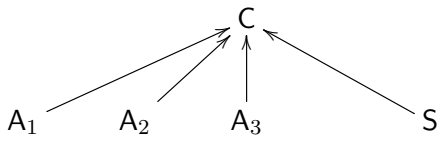


Figure 4.1: Sybil attack

Solution Our solution is binding protocols. These protocols provide a secure way of allocating identities. In addition, these identities can be easily updated. In summary, we want to provide a better identity system for personal networks.

4.2.2 Binding Protocols

Distributed binding Using distributed binding protocol HISP-B[3] I, each node receives one unique address from C. Suppose each node is allocated a temporary address for protocol communications. Note that the threat model and assumptions are

- HCCs are NSB channels
- The user and all the nodes are trustworthy.
- group size is inputted via a NSB HCC.
- The notification/verification signal (e.g. LED alarm signal that should be read by human users) is transmitted over a NSB HCC.
- Button pushing procedure is a NSB HCC.

The message flow is shown in Protocol 4.4. The details are explained below.

Commit phase step 1. (a) *bp*: user inputs the group size gs on C, and pushes a button on each node. (b) C generates its address $address_{\mathsf{C}}$, a long random number N_{C}, and a hash commitment $HC_{\mathsf{C}} = hash(N_{\mathsf{C}}, \mathsf{C})$. (c) C sends S_i *message 1* including its identity C, $address_{\mathsf{C}}$, gs, and HC_{C} over wireless radio channels.

[3]B for binding

Protocol 4.4 (HISP-B I)**.**

1. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : \mathsf{C}, address_{\mathsf{C}}, gs, HC_{\mathsf{C}}$
2. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} \wedge \mathsf{S}_j : \mathsf{S}_i, address_i$
3. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : N_{\mathsf{C}}$
4. $\forall \mathsf{S}_i \wedge \mathsf{C} \Longrightarrow \text{User} : D$

Commit phase step 2. (a) After message 1 has been received, each S_i broadcasts *message 2* including its identity S_i and address $address_i$ over wireless radio channels. (b) C counts the number of received message 2. If the number is $gs-1$, it ceases to accept new message 2, and flashes its LEDs. (c) Each S_i also counts the number of received commitment messages: message 1 and 2; if the number is $gs-1$, it ceases to accept new commitment messages, and it flashes its LEDs. (d) *cp*: after all the nodes have flashed their LEDs, the user informs C to release message 3.

Reveal phase step 3. (a) *rp*: C sends the opening message *message 3* including N_{C} to each S_i over wireless radio channels. (b) S_i verifies HC_{C}. If the verification fails, S_i aborts and informs the user.

Reveal phase step 4. (a) S_i and C compute $D = digest(N_{\mathsf{C}}, gs, \mathsf{C}, address_{\mathsf{C}}, \sum(\mathsf{S}_i, address_i))$. (b) Each node transfers *message 4* including its digest value D to the user via human-controlled NSB channels. (c) *ep*: if all the digest values are the same, the user pushes the confirmation button on each node. In this case, all the nodes have received correct addresses.

The distributed model may lead to address confliction; and if confliction happens, the node needs to regenerate a new address. In the distributed binding protocol, the confliction free probability of gs group members with k possible addresses is as follows (C_k^{gs} is combination, and H_k^{gs} is combination with repetitions):

$$p = \frac{C_k^{gs}}{H_k^{gs}} = \frac{\binom{k}{gs}}{\binom{gs+k-1}{gs}}$$

Assume that the address lengths are 8, 10 and 12 bits respectively. The confliction free rates are shown in Figure 4.2. The results show that 8 bits is good enough for five nodes

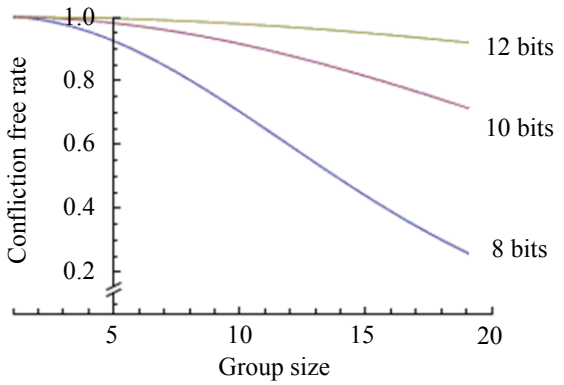


Figure 4.2: The confliction free rate of 8, 10 and 12 bits names. When group size is 5, the 8 bits addresses have a successful rate higher than 90%.

or less, so it is enough for personal networks in most cases.

Centralized binding A centralized binding works differently. In a centralized model, the coordinator node generates corresponding addresses for all the slave nodes. Note that the threat model and assumptions are

- HCCs are NSB channels
- The user and all the nodes are trustworthy.
- group size is inputted via a NSB HCC.
- The notification/verification signal (e.g. LED alarm signal that should be read by human users) is transmitted over a NSB HCC.
- Button pushing procedure is a NSB HCC.

The message flow is shown in Protocol 4.5. The details are explained below.

Protocol 4.5 (HISP-B II)**.**

1. $\mathsf{C} \longrightarrow \forall\, \mathsf{S}_i : \mathsf{C}, address_{\mathsf{C}}, \sum(address_i), hash(N_{\mathsf{C}}, \mathsf{C})$
2. $\mathsf{C} \longrightarrow \forall\, \mathsf{S}_i : N_{\mathsf{C}}$
3. $\forall\, \mathsf{S}_i \wedge \mathsf{C} \Longrightarrow \text{User} : D$

Commit phase step 1. (a) *bp*: the user inputs the group size *gs* on C, and pushes a button on each node. (b) C generates its address $address_{\mathsf{C}}$, addresses for slave nodes $\sum(address_i)$,

a long random number N_{C}, and a hash commitment $HC_{\mathsf{C}} = hash(N_{\mathsf{C}}, \mathsf{C})$. (c) C sends S_i *message 1* including C, $address_{\mathsf{C}}$, $\sum(\mathsf{S}_i, address_i)$, and HC_{C} over wireless radio channels. (d) After message 1 has been received, each S_i flashes LEDs; from now on, S_i does not accept new message 1. (e) *cp*: user verifies that each S_i has flashed, and checks the number of these slave nodes is $gs - 1$. If the verification succeeds, the user informs C to release the opening message (message 2) in step 2.

Reveal phase step 2. (a) *rp*: C sends the opening message *message 2* including N_{C} to each S_i over wireless radio channels. (b) S_i verifies HC_{C}. If the verification fails, S_i aborts and informs the user.

Reveal phase step 3. (a) S_i and C compute $D = digest(N_{\mathsf{C}}, \mathsf{C}, address_{\mathsf{C}}, \sum(\mathsf{S}_i, address_i))$. (b) Each node transfers *message 3* including its digest value D to the user via human-controlled NSB channels. (c) *ep*: if all the digest values are the same, the user pushes the confirmation button on each node. In this case, all the nodes have received correct addresses.

4.2.3 Understanding binding protocols

Sybil attack elimination These binding protocols eliminate sybil attacks. In sybil attacks, one malicious node A illegitimately uses multiple identities. In our binding protocols, the group size gs is controlled by the user, thus if A wants to use x_{A} illegal identities, it must stop x_{A} legal nodes from joining in the group. A has the following strategies:

- A may prevent x_{A} legal nodes from running the protocols. However, in our binding protocols, the user needs to check the digest outputs in each node at the final step. Thus, this strategy will not succeed.

- A partitions the network. In this case, the final digest outputs will not be the same. This attack also fails.

Therefore, identities including addresses can be allocated and updated without worrying about sybil attacks.

Applications The potential application is discussed as follows. A person wears several slave nodes in order to measure heartbeat, etc. The personal network can be simply configured as a star network with a local controller node dealing with data aggregation. A gateway at home sends the aggregated data to a healthcare centre. A mobile phone is also used to check the data at home. When the person goes out, the mobile phone can also act as the gateway for transferring sensor data. Since this always costs money and consumes battery power, if the person arrives at his office or another public place, the personal network looks for other possible gateways.

Our identity system is suitable for such a user case. The slave nodes need to submit data via different gateways. The gateways only work as a tunnel from the slave nodes to remote applications, so secure identity exchange is enough to make sure that a trusted link is established (keys may not be necessary). This procedure must be dynamic, because the gateways may not have users' information previously.

Additionally, there may be slave nodes installed at home, in the office or other places. The personal network could receive information from these places to provide context-aware services. In this case, their sensor data is always public, so secure and dynamic identity exchange is enough.

4.3 Chapter Summary

This chapter contains two HISP use cases. These use cases show that HISPs are useful building blocks of security mechanisms in wearable personal networks.

The first is the ECDH version of HISPs, eliminating man-in-the-middle attacks against ECDH. ECDH is currently used for key distributions in TinyECC [70], one of the main public key cryptography libraries for nodes running TinyOS. However, ECDH can bring about serious problems, because it is vulnerable to man-in-the-middle attacks. Suppose attacker A initiates such an attack, A can eavesdrop any data in the network. In addition, A can generate fake data, which may for example cause wrong treatments. Hence, ECDH version of HISPs are meaningful for personal networks.

The second is identity binding protocols, which can eliminate sybil attacks. Identity (ID)

systems of personal networks mainly focus on the relationship between nodes and identities. Binding is a mapping between nodes and identities. However, in a sybil attack [79], a single malicious node illegitimately uses multiple identities. This makes other nodes believe in a wrong topology, which can significantly interrupt media access control sub-layer and routing layer protocols. Wrong sensor data will then be collected by the coordinator node. In order to provide a better identity system, we have proposed several binding protocols. These protocols provide a secure way of allocating identities. In addition, these identities can be easily updated.

To conclude, HISPs are useful security protocols in wearable personal networks. They bring new possibilities; and they can help us to solve some real-world problems, for example, ECDH MITM attacks and Sybil attacks.

Chapter 5

Visible Light Communications and Security Protocols

HVSPs[1] are protocols designed to provide security in wearable personal networks using HVLCs[2]. The concept design is as follows. In wearable personal networks, wireless radio channels are used for data communications, and properly selected HVLCs are used to protect the data.

HVSPs use HVLCs to compare LED flash patterns, and thus the human burden is reduced, which is their main advantage compared with HISPs. In addition, for similar reasons, HVSPs can be used in small nodes. Also, HVSPs are flexible because they do not require pre-distributed secrets.

Contents

[1] H for human-controlled, V for visible light communication, SP for security protocol
[2] H for human-controlled, VLC for visible light communications

5.1 Visible Light Communications

In this section, we will introduce (1) background, and (2) channel establishment techniques.

5.1.1 Background

Visible light communications (VLCs) are communication channels that use visible light as transmission media. It is well known that electromagnetic radiation is classified by wavelength into radio, microwave, infrared, visible light, ultraviolet, X-rays, and gamma rays. Visible light is electromagnetic radiation roughly between 380 nm and 780 nm.

VLC history and recent development Historically, VLCs have already been used in many scenarios. For example, lamps are used in lighthouses; flag signals are used for communications between ships; traffic signals are used for traffic control; and a sunlight-based audio signal transmission system was designed by Alexander Graham Bell in the 19th century.

In last few years, there has been a growing interest in VLCs [46]. From 2004 to 2007, there were VLC demonstrations including a VLC from a LED to mobile phones, and a VLC from a LED-backlit TV to a PDA. From 2007 to 2009, several standards were proposed by Japan, EU and USA, including a Visible Light ID System Standard. In 2010, VLCs were developed among TV, PDA, mobile phone, and so on; a VLC-based indoor positioning system was proposed; and VLC speed achieved 500 Mbps. In 2011, the IEEE 802.15 Working Group for Wireless Personal Area Networks completed draft 5c of a physical layer and medium access control sublayer standard for VLCs.

Drivers of VLCs VLCs have many advantages, and these are the drivers of VLCs. Firstly, more and more LEDs are used today. Thus, it is possible to establish low-cost VLCs everywhere using existing devices. Secondly, VLCs pose no known health problems, hence they are ideal for personal networks associated with human users. Thirdly, the visible band is free from frequency regulation and wireless radio interference. Thus, they are ideal for wireless radio crowded or restricted environments, for example, aircraft cabins and petrochemical industries.

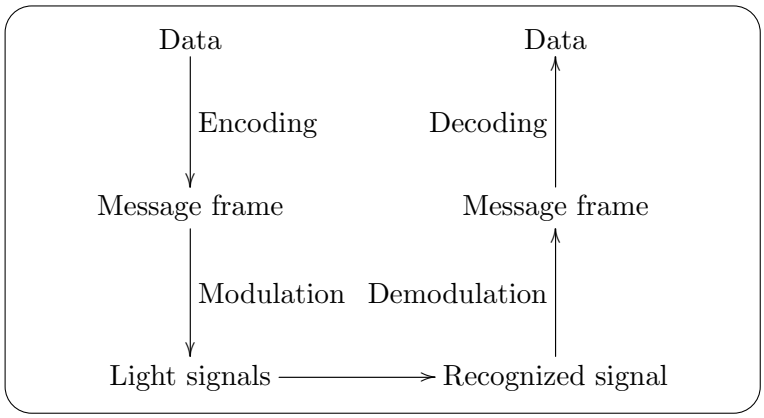


Figure 5.1: VLC establishment procedure

Preamble	Delimiter	Sender ID	data

Figure 5.2: Message frame

Additionally, VLCs are relatively more secure than wireless radio (used in sensor networks) and infrared. Firstly, visible light does not penetrate walls, but wireless radio generally does. In this case, VLC signals are relatively difficult to be eavesdropped when attackers are outside the room. Secondly, users can see the sources of VLCs, and users can perceive interferences during the communication procedure. This is not true for wireless radio and infrared. Thus, communication authenticity, integrity and freshness of VLCs are relatively easily guaranteed.

5.1.2 VLCs Establishment

VLC devices VLCs are established using light sources such as transmitters and light sensors (including cameras) as receivers. Other optical devices, for example filters, are also useful in some scenarios.

LEDs, which are semiconductor light sources, are becoming popular. This is the result of the following advantages: short switching times, ideal as a communication device subject to frequent on-off with a simple driver circuit, they are thus easily populated onto nodes; high intensity light with low power consumption; emit light of an intended colour without the use of colour filters; and have no health hazards.

Establishment procedure VLCs are generally established as in Fig. 5.1; and the procedural steps are explained as follows.

1. Data is encoded into a message frame. An example of a message frame is shown in Fig. 5.2. A preamble (1010) is used for communication synchronization[3]; and a delimiter (11) denotes the end of the preamble.

2. Each bit of the frame is modulated by the modulator, and then transmitted using LED light. Modulation is the process of conveying a message signal inside carrier signal that can be physically transmitted; and the device that performs modulation is called a modulator. We will use on-off keying (OOK), which represents digital data as the presence or absence of a carrier signal. In its simplest form, if the bit is '1', LED is turned on; if the bit is '0', LED is turned off.

3. LED light signals are received by light sensors.

4. Received LED signals are demodulated to the message frame by the demodulator, which is a device that performs the inverse operation of modulation.

5. The message frame is decoded to data.

VLCs in personal networks VLCs can be used in personal networks. Both light transmitters and light receivers can be found: most nodes have *LEDs*, for example, Tmote Sky nodes, MICAz nodes and MICA2 nodes are embedded with several LEDs; most current mobile phones have *flash lights*; many slave nodes and most current mobile phones are embedded with *light sensors*; and most current mobile phones and personal computers are equipped with *cameras*.

Example (VLCs in personal networks). Two types of VLCs are shown in Fig. 5.3. The first type: LEDs on the mobile phone flash and send out light signals that are depicted as a series of white and black dots (10101100001110101010101010); light sensors on the slave

[3]Synchronization problems are critical [59]. The receiver has to learn the frequency and the phase of the signal; the receiver must determine both the symbol duration as well as the start and the end of symbols; the receiver of a frame must be able to detect where the frame starts and where it ends; and receiver must compensate the clock drift.

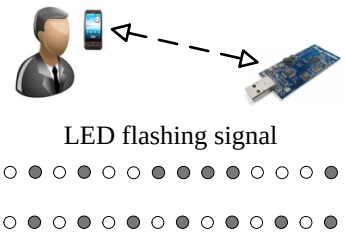
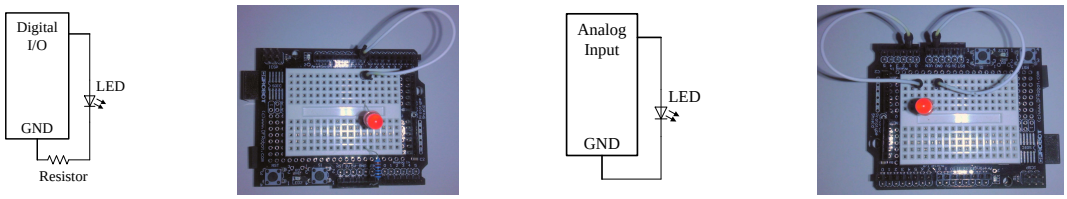


Figure 5.3: Example VLCs in personal networks



(a) (b) (c) (d)

Figure 5.4: LED as transmitter and receiver.

node receive these light signals. The second type: LEDs on the slave node flash and send out light signals; a camera on the mobile phone receives these light signals.

5.2 LED-Sensor Channel

LED-sensor channel (LS) is one type of VLC. We will explain (1) transmitters, (2) receivers and filters, (3) channel implementation, and (4) some experiment results.

5.2.1 Transmitters

LSs use LEDs as transmitters. One example of LED transmitter that we have made is shown in Fig. 5.4(a) and Fig. 5.4(b). The LED is directly connected to a digital I/O port, and there is a load resistance in series with the LED.

LED radiation intensity r can be normally modelled using the Lambertian radiation pattern. Suppose m is the order of Lambertian emission; and ϕ is the irradiance angle, r can be modelled as follows [46]

$$r(\phi) = \frac{(m+1)}{2\pi} \cos^m(\phi)\,,\ \phi \in [-\pi/2, \pi/2]. \tag{5.1}$$

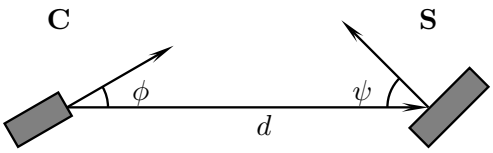


Figure 5.5: LS body area channel model.

5.2.2 Receivers and Filters

Receivers The LS receiver is a electronic device generally containing a light sensor that converts incident visible light into an electrical signal. Light sensors can be found in many nodes. For example, the Tmote sky node is normally populated by two photodiodes from Hamamatsu Corporation, and both MTS101CA and MTS510CA sensor boards used on mica nodes are populated with photo resistors. We have also made a LS receiver using one LED as a light sensor. This example of receiver is shown in Fig. 5.4(c) and Fig. 5.4(d).

Wavelength sensitivity and filters Wavelength sensitivity is an important factor of light sensors. It is the ratio of the energy generated by a light sensor to the energy of incident light. For example, the light sensor S1087 that is embedded on the Tmote sky node is sensitive to light between 300nm and 800nm; and light sensor S1087-01 that is also embedded on Tmote sky is sensitive to the entire visible spectrum and to infrared. As we can see here, some light sensors also sensitive to invisible light.

In order to eliminate possible attacks using invisible light, optical filters can be used. They can transmit light in a particular range of wavelengths while blocking the remainder.

Physical model Suppose the relation between the receiver and the transmitter is shown in Fig. 5.5. The DC gain of the receiver [46] is

$$h_0 = \frac{ar(\phi)\cos(\psi)T(\psi)g(\psi)}{d^2}\,,\ \psi \in [0, \Psi]. \tag{5.2}$$

Related parameters are introduced as follows. The area that collects the radiation is a. The distance between C and S is d, where $d \gg \sqrt{a}$. The incident angle ψ is smaller than the field of view Ψ of the light sensor on S; Ψ is the angular extent of coverage. The gain of the optical filter is $T(\psi)$; and the optical gain of the nonimaging concentrator is $g(\psi)$.

For the line-of-sight case, if c is the speed of light, the impulse response $h(t)$ of the receiver at time t can be approximated using the Dirac delta function $\delta()$ as follows [46]

$$h(t) = h_0 \times \delta(t - \frac{d}{c}), \tag{5.3}$$

where $\delta(t - d/c)$ represents the signal propagation delay.

The received signal $y(t)$ of S is thus as follows [46]

$$y(t) = \gamma(x(t) \otimes h(t)) + n(t), \tag{5.4}$$

where γ is the light sensor sensitivity; $x(t)$ is the input signal in C; $\otimes$ denotes convolution; and $n(t)$ is the additive white Gaussian noise, which is typically shot noise caused by ambient light or receiver preamplifier noise.

Raw method of signal detection From equation 5.4, we can get a raw method of LS signal detection. This method is described as follows.

1. Receiver decides a threshold thd before it starts to receive $x(t)$. Suppose $n(t)$ is the detected noise in a period $t \in [0, T]$, thd should fulfil the following inequality.

$$thd > \max(n(t)) \tag{5.5}$$

2. Receiver continuously reads voltages v. If $v - thd > 0$, it outputs 1; otherwise, it outputs 0.

5.2.3 Channel Implementation

In summary, LS is established as follows. Suppose we want to transmit a digest output (16 bits). The LED signal generation procedure is shown in Procedure 5.1. The receiving procedure is shown in Procedure 5.2. If we want to transmit a hash output (160 bits), we can separate it into 10 frames, and then these frames are transmitted using Procedure 5.1 and Procedure 5.2.

Procedure 5.1 LED signal generation process

1: Data is encoded into a message frame. The frame consists of preamble (1010), delimiter (11), sender ID (5 bits) and data (16 bit).
2: Each bit of the frame is represented as the presence or absence of LED light, which is the OOK modulation.

Procedure 5.2 LED signal receiving process

1: Receiver computes a threshold thd.
2: It continuously reads voltages v with high frequency. If $v - thd > 0$, it outputs 1; otherwise, it outputs 0.
3: It starts a synchronization loop when 1 is detected. In the loop, the frequency becomes the blinking frequency f. It continuously reads voltages v. If $v - thd > 0$, it outputs 1; otherwise, it outputs 0. If 101011 is detected, it goes to step 4, otherwise, it goes back to step 2.
4: It reads a sequence of voltages $v(t)$, where time $t = (seqn - 0.5)/f + \delta$, $seqn \in [1, 21]$ (the payload length is 21 bits), and δ is a small number ($\delta < 0.5$) in order to compensate the clock drift.

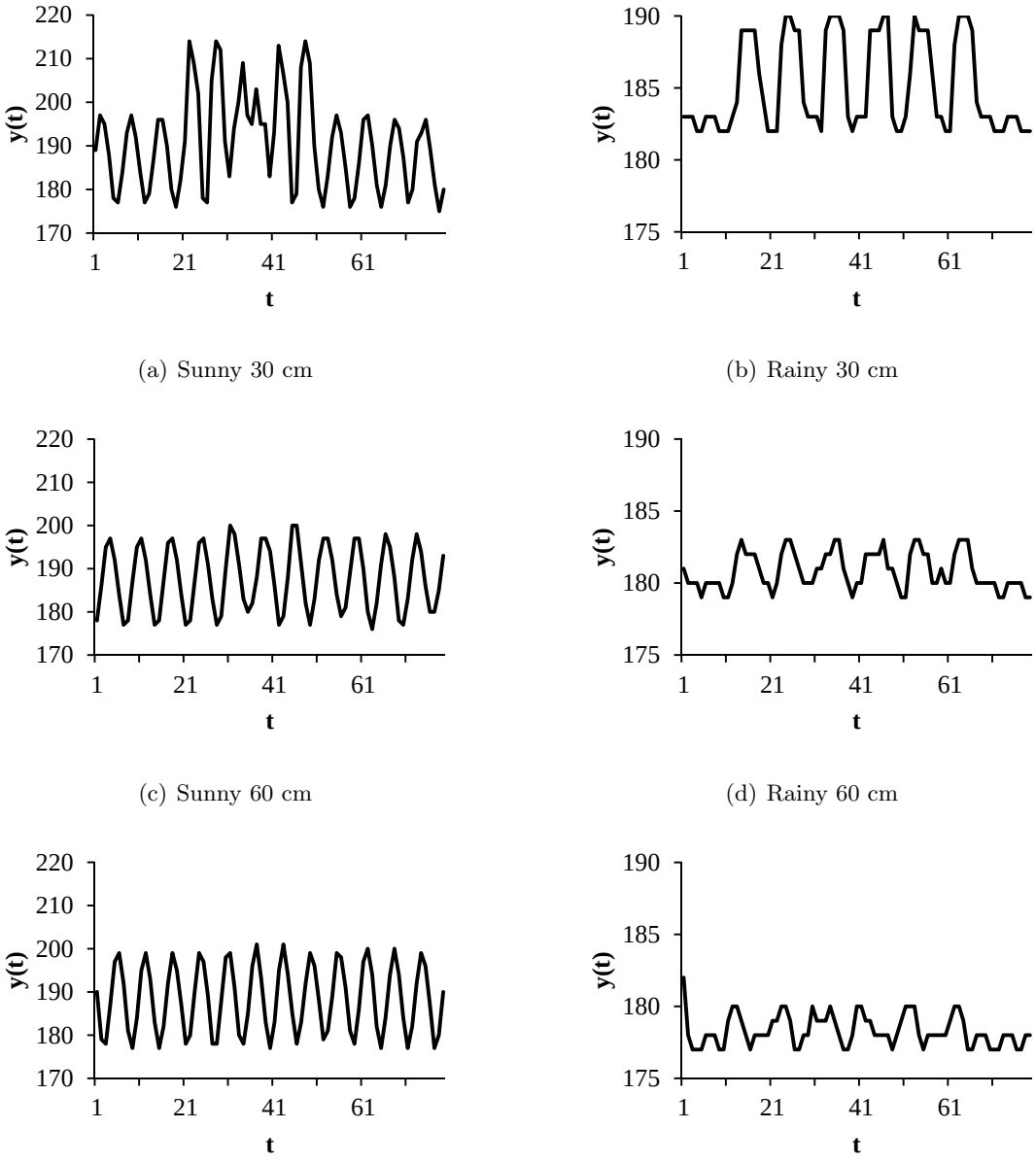
5.2.4 Experiment

The first experiment is a 200-bit message transmission. The LS transmitter is placed right above the receiver around 10cm in a room with no interference light sources. We tested 10 times. All the messages are correctly received within 1 second.

In the second experiment, we tested the parameters that influence LSs. There are many parameters based on equations from 5.1 to 5.4. However, in many practical cases, we cannot change the light source and light sensor; thus the main factors are distance d and illuminance condition (this condition largely decides $n(t)$). We tested LSs with different distances d and various illuminance conditions. The light source is two flash lights on a mobile phone; and the receiver is based on a LED, which was previously shown. Fig. 5.6(a), Fig. 5.6(c), and Fig. 5.6(e) are the results of three tests in a sunny afternoon; Fig. 5.6(b), Fig. 5.6(d), and Fig. 5.6(f) are the results of three tests in a rainy afternoon. Below are the results.

- In the sunny afternoon, when $d = 30cm$, receiver can distinguish $x(t)$ from noise $n(t)$; however, when $d = 60cm$ or $d = 90cm$, the receiver cannot.

- In the rainy afternoon, when $d = 30cm$ and $d = 60cm$, receiver can distinguish $x(t)$ from noise $n(t)$; however, when $d = 90cm$, the receiver cannot.



(a) Sunny 30 cm

(b) Rainy 30 cm

(c) Sunny 60 cm

(d) Rainy 60 cm

(e) Sunny 90 cm

(f) Rainy 90 cm

Figure 5.6: LS channel tests in a sunny afternoon and in a rainy afternoon.

Thus, when d increases, the quality of the channel decreases; and when the ambient light intensity is low, the quality is better. In our scenario, we suggest that the user should place the LEDs as close to the receiver as possible.

(a) Preamble signal I (b) Preamble signal II (c) Data signal I (d) Data signal II

Figure 5.7: RGB LED images

5.3 LED-Camera Channel

LED-camera channels (LCs) are established using LEDs and cameras, which use image processing techniques. This section includes (1) image representation, (2) morphological operations, (3) object representation, (4) channel implementation, and (5) some experiment results.

5.3.1 Image Representation

There are many representation methods of images and pixels [47]. We have used several representation models in LC establishment procedures. These models are introduced below.

Representation models In the RGB model, each pixel is represented by three intensities: red R, green G, and blue B. In Fig. 5.7, we show four LED images in a RGB model. The first two are preamble signals. The second two are data signals.

A gray scale model is also named an intensity model. In this model, each pixel in an image is represented by an intensity value p. Normally,

$$\begin{cases} p \in [0, 255], & \text{each pixel is represented using an 8-bit number;} \\ p \in [0, 65535], & \text{each pixel is represented using a 16-bit number.} \end{cases} \tag{5.6}$$

In the binary model, each pixel has only two logical values: 1 and 0. The pixel is named the foreground pixel or background pixel based on the pixel value p as follows:

$$p = \begin{cases} 1, & \text{foreground pixel;} \\ 0, & \text{background pixel.} \end{cases} \tag{5.7}$$

(a) Preamble signal I (b) Preamble signal II (c) Data signal I (d) Data signal II

Figure 5.8: LED signal recognition using RGB model.

(a) (b) (c) (d)

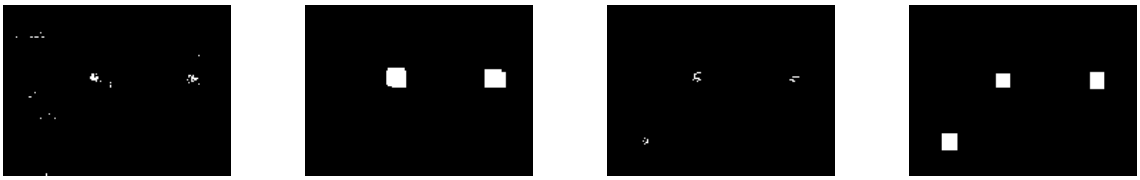
Figure 5.9: Find positions of LEDs using gray and binary models.

Usages The RGB model and binary model can be used to recognize LED signals. We filter the RGB images using a threshold in the first step, for example red colour intensity $R > 253$; and in the second step, we change the filtered images to binary images. With the two steps, we can change the subfigures in Fig. 5.7 to subfigures in Fig. 5.8. As we can see, most irrelevant parts of the subfigures are removed. After morphology operations, which will be introduced later, the locations of LEDs that are turned on can be found.

Using gray and binary models, we can also find the positions of LEDs. In Fig. 5.9, we show the results. Firstly, we compute the difference between Fig. 5.7(b) and Fig. 5.7(a). Secondly, we represent the difference as a gray image, which is Fig. 5.9(a). Thirdly, we set a threshold and change Fig. 5.9(a) to a binary image Fig. 5.9(b). Similarly, we can get Fig. 5.9(c) and Fig. 5.9(d) from the difference between Fig. 5.7(c) and Fig. 5.7(a).

5.3.2 Morphological Operations

Morphological operations can extract topological and geometrical structures [47]. There are two basic morphological operations: dilation, which thickens the object; and erosion, which shrinks the object. They are always used together in practical applications. As we can see in Fig. 5.10, morphological operations can remove noise dots: (a) and (c) are images before morphological operations, while (b) and (d) are images after the operations.



(a) (b) (c) (d)

Figure 5.10: LED recognition results after morphological operations

5.3.3 Object Representation

An object in a binary image is a group of connected pixels. There are normally two kinds of object: 4-connected and 8-connected [47]. Generally speaking, in the 4-connected objects, two foreground neighbour pixels $p = (x, y)$ and q are connected if $q \in N_4(p)$, where

$$N_4(p) = \{(x+1, y), (x-1, y), (x, y+1), (x, y-1)\}. \tag{5.8}$$

Similarly, in the 8-connected objects, $q \in N_8(p)$, where

$$N_8(p) = N_4(p) \cup \{(x+1, y+1), (x-1, y+1), (x+1, y-1), (x-1, y-1)\}. \tag{5.9}$$

In Fig 5.10, each isolated white dot is an object.

5.3.4 Channel Implementation

LC is established as follows. The LED signal generation process is the same as that in the LS. In addition, the LED signals are automatically recognized by a fixed camera using Procedure 5.3. Finally, after demodulation and decoding, the receiver gets data.

Alternatively, we can use colour information to identify LEDs. This method can tolerate short-distance movements of the camera; however, it has a pre-requisite condition: the colour of the LEDs is different from the colour of the nodes. Fortunately, most nodes fulfil this condition.

The third LED signal recognition process uses three LEDs in each node. Two LEDs are used for synchronization. The third LED is used for data transmission. From our experience, this method is the best one regarding the transmission speed.

Procedure 5.3 LED signal recognition process

1: Camera takes a photo when all LEDs are off, say p_f; and a photo when all LEDs are on, say p_n.
2: Find locations of LED centres locs based on $p_n - p_f$: convert image $p_n - p_f$ to binary image p_{bw}; applies morphological operations to p_{bw}; label connected components in p_{bw}; locations locs are the centres of mass of the regions in each connected components;
3: Camera takes a sequence of photo images $p(t)$, where $t = (seqn - 0.5)/f + \delta$, f is the blinking frequency, $seqn \in [1, 27]$ (the message frame length is 27 bits), and δ is a small number ($\delta < 0.5$) in order to reduce the influence of clock drift.
4: In each $p(t)$, for LED located in loc, if $v(loc, p(t)) > (v(loc, p_n) - v(loc, p_f))/2$, the LED is on, otherwise, the LED is off. $v(loc, p(t))$, $v(loc, p_n)$ and $v(loc, p_f)$ are the values of the pixel located at loc in $p(t)$, p_n and p_f.

5.3.5 Experiments

We did several LC tests using Matlab version 7.3 installed in a laptop (Intel Core 2 Duo processor T7500 2.2GHz, 2GB DDR2 RAM, Windows 7 Home Premium), Microsoft LifeCam VX-1000, and red LEDs. The tests are introduced below.

Recognition of signals from red LEDs in three Tmote sky nodes. The signals from the three LEDs differ, the signals are not synchronized, and the flashing frequency is 2 bps. The recognitions are successful in all the tests.

Successful recognition rate of signals from one LED using Procedure 5.3. When the flashing frequency is 2 bps, we successfully recognize all 50 random 27-bit strings. When the flashing frequency is 4 bps, the computer cannot correctly recognize the strings because of clock drift in the slave nodes.

Recognition speed. When we use Procedure 5.3, the stable flashing frequency is 2 bps, and thus the LED flashing time is about 14 seconds; in addition, the average computing time for recognition is less than 10 seconds. However, when we use the third LED recognition method, the speed can reach 10 bps; thus the 27-bit message can be transmitted in around 3 seconds.

Also, LEDs were recognized successfully when we changed the distances between the LEDs and the camera from 30 cm to 60 cm. In addition, the processing speed was faster when a low-resolution mode was used.

5.4 Human-Controlled VLCs

This section introduces HVLCs. It includes (1) related definitions, (2) security analysis, and (3) networking characteristics.

5.4.1 Related Definitions

Definition 5.1 (Human-controlled VLC (HVLC)). HVLC is VLC controlled by human users. Human users can determine (1) which node will be involved in a certain communication procedure; (2) when the node will be involved in a certain communication procedure; (3) when the information will be transmitted. Besides, users can perceive and prevent interference during the communication.

Definition 5.2 (Human-controlled LED-sensor channel (HLS)). HLS is LS controlled by human users.

Definition 5.3 (Human-controlled LED-camera channel (HLC)). HLC is LC controlled by human users.

HVLCs are usable in personal networks. First of all, VLCs, the foundation of HVLCs, can be used in personal networks; this has been discussed previously. Additionally, nodes are worn by human users; it is easy for the users to control these channels.

5.4.2 NSB HVLCs

One of the most useful HVLCs is a NSB HVLC. We especially define this channel in Definition 5.4. For convenience, we do not distinguish NSB HVLCs from HVLCs in the rest of this thesis.

Definition 5.4 (NSB HVLC). A NSB HVLC is a HVLC, and should be modeled as a NSB channel from a security standpoint.

Generally speaking, the channels that we have implemented are NSB channels. This is because HVLC offers several security advantages over existing wireless radio technologies:

- Visible authenticity. You can see that the data comes from the desired authentic source. Wireless radio and other invisible communications, for example, IR, do not

have this advantage.

- Line-of-sight communication. Light travels in a straight line; and light does not pass through walls.
- Conditional confidentiality. Since VLC does not pass through walls, eavesdropping can be prevented in some scenarios.

Threat model and assumptions Threat model. HVLCs that will be used in our protocols are assumed to be NSB channels. In other words, attackers can overhear messages over HVLCs, however they cannot block, modify, fake, replay, or delay messages. This threat model is based on assumptions elaborated below.

Assumption 1: users are trustworthy. It means that users will not attack nodes or protocols. Also, they will follow the instructions in protocols.

Assumption 2: nodes are trustworthy. In other words, there are no malicious programs that will change messages transmitted or received

Assumption 3: users can found and prevent the following attacks: the attacker A blocks, modifies, fakes, replays, or delays messages over HVLCs.

Firstly, during the communication procedure, in order to disrupt the VLCs between two nodes, the attacker A must influence the light intensity or colour perceived by the light sensor or camera. There are generally two strategies.

1. A blocks the legal visible light beam. However, users can see that the legal visible light beam is blocked.
2. A changes ambient light in order to "blind" the light sensor or camera. However, users can also see that the ambient light is changed.

Therefore, users can found and prevent the following attacks: the attacker A blocks or delays messages over HVLCs.

Secondly, in order to change messages, replay messages, or initiate a fake conversation, the attacker A has two strategies.

1. A interferes the legal light beam by using another fake visible light beam or ambient light. However, the attack light beam and changes of ambient light can be seen by users.

2. A uses a fake node instead of a legal node. This is impossible since the nodes are controlled by users.

Therefore, users can found and prevent the following attacks: the attacker A modifies, fakes, or replays messages over HVLCs.

Assumption 4: attackers can overhear messages over HVLCs.

A can eavesdrop HVLC messages. For example, A can use professional high frame rate video cameras to capture the LED flashing signals [73].

User expectations (1) When users are running protocols, especially when messages are transmitting over HVLCs among nodes, users should be alert to the following scenarios: a) someone uses a light source and points at these nodes; b) the ambient light strength is changed suddenly; c) someone tries to block the light from these nodes. (2) Users should update operating systems or even use anti-virus software in order to make sure that their nodes are trustworthy. (3) Users should not allow other people to use their nodes unless a) they are trustworthy, and b) this is necessary. (4) If possible, users should put nodes as close as possible to prevent attacks against HVLCs.

5.4.3 Networking characteristics

The networking characteristics of HLSs and HLCs can significantly influence the protocol design. Thus, we analyze their networking characteristics below.

HLSs have the following characteristics:

- Transmission rate can be relatively high.

- LED and light sensors must be in a certain spatial relationship (line-of-sight).

- In many practical cases, they are one-to-many channels. C can sends data to each S via HLSs.

- The processing capabilities of nodes are high.

HLCs have the following characteristics:

- LED and camera must be in a certain spatial relationship (line-of-sight).
- The speeds of many HLCs are low. Transmission rate depends on the maximum flashing speed of LED and the maximum sample rate of the camera. Most simple cameras do not have a high video frame rate, thus speed is low.
- In many cases, they are many-to-one channels. Cameras are associated with C. They can collect data from each S via HLCs.
- The processing capability of C is high.

Example. Suppose the transmission rate of HLC is 10 bps. The hash output H is usually longer than 160 bits. The length of message frame that contains 160-bit hash is 172 bits. It takes around 17 seconds to transmit the whole frame, which is very slow. However, if we use a HLS that supports 200 bps communications, the 172-bit message can be transmitted within 1 second. □

5.5 HVLC-based Security Protocols

This section introduces HVSPs. It includes (1) HVSPs, and (2) example protocols.

5.5.1 HVSPs

In HVSPs, we usually use NSB HVLCs in order to compare a series of LED flash patterns. We can verify the data transmitted via other channels, i.e. wireless radio channels, and provide security in personal networks. A definition of HVSP is given below.

Definition 5.5 (HVLC-based security protocol (HVSP)). A HVSP is a protocol that provides security in personal networks based on the security properties of HVLCs.

Definition 5.6 (HLS-based security protocol). A HLS-based security protocol is a HVSP using HLSs.

Definition 5.7 (HLC-based security protocol). A HLC-based security protocol is a HVSP using HLCs.

Since HLSs usually support high speed one-to-many communications, they can broadcast several digest values in a short time. Meanwhile, since HLCs usually support low-speed many-to-one channels, they can receive several digest values in a short time. Therefore, instead of a group commitment scheme, we can use a parallel commitment scheme that is more suitable for HVSPs. This scheme is defined below.

Definition 5.8 (Parallel commitment scheme). A parallel commitment scheme allows C and each S_i to commit to a secret value Θ_i in parallel; it also allows these values to be revealed and checked later. It consists of two phases: a commit phase and a reveal phase. In addition, the scheme holds hiding property and binding property. Besides, there are four significant time points.

- Commit phase. Each pair (C and S_i) exchanges commitment messages. Using these messages, C and S_i cooperatively choose a secret value Θ_i. Note that group size should be explicitly or implicitly checked *only in* C in this phase.

- Reveal phase. Each pair (C and S_i) exchanges opening messages. Using these messages, Θ_i is revealed and checked.

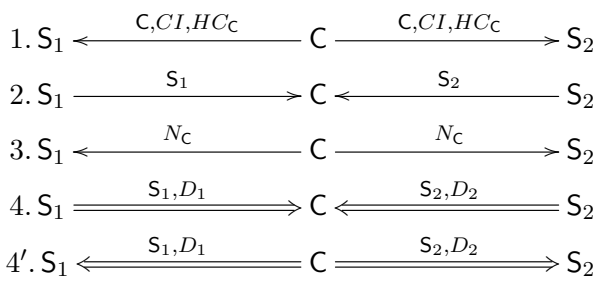
- Hiding property. Θ_i is a value hidden from the receiver, and even from the sender during the commit phase.

- Binding property. During the reveal phase, Θ_i must be the only value that each pair can compute and that can be validated.

- Time points. The beginning point is named *bp*, and the end point is named *ep*. The time that all the nodes cease to accept new commitment messages is named commit point *cp*. The time that a node (might be A) knows the value of Θ_i is named reveal point *rp*.

Certainly, high speed HLSs can also support hash-based security protocols. We will see this in the next chapter.

Protocol 5.1 (HVSP-CI)**.**



5.5.2 Example HVSPs with A Parallel Commitment Scheme

HVSP-CI HVSP-CI is one example of HVSP with a parallel commitment scheme. Suppose there is a personal network that consists of one coordinator C, and two slave nodes S_1 and S_2. HVSP-CI helps C to deliver instructions CI securely. The message flow is shown in Protocol. 5.1. The details are explained below.

Commit phase step 1. (a) *bp*: the user inputs the group size $gs = 3$ on C, and pushes a button on each node in order to initiate the protocol. (b) C generates a long nonce N_{C} and a hash commitment $HC_{\mathsf{C}} = hash(N_{\mathsf{C}}, \mathsf{C})$. (c) C broadcasts *message 1* including its identity C, instructions CI, and commitment HC_{C} over wireless radio channels. (d) After message 1 has been received, each slave node ceases to accept any new message 1.

Commit phase step 2. (a) After step 1.d, each slave node broadcasts *message 2* including its identity (S_1, S_2) over wireless radio channels. (b) After step 2.a, each slave node flashes its LEDs. (c) C counts the number of received message 2. If the number is $gs - 1 = 2$, C ceases to accept any new message 2, and flashes its LEDs. (e) *cp*: if all the nodes have flashed their LEDs, the user informs C to release the opening message (message 3) in step 3.

Reveal phase step 3. (a) *rp*: C broadcasts the opening message *message 3* including N_{C} over wireless radio channels. (b) Slave nodes verify HC_{C}. If the verification fails, they abort and inform the user.

Reveal phase step 4. (a) S_1 broadcasts *message 4* including its identity S_1 and the digest output D_1 via HLCs, where $D_1 = digest(N_{\mathsf{C}}, \mathsf{C}, \mathsf{S}_1, CI)$. (b) Meanwhile, S_2 broadcasts *message 4* including its identity S_2 and the digest output D_2 via HLCs, where $D_2 =$

$digest(N_{\mathsf{C}}, \mathsf{C}, \mathsf{S}_2, CI)$. (c) *ep*: C verifies the received digest value. It informs the user of the verification results.

Alternative reveal phase step 4. (a) C sends S_1 *message 4* including the target identity S_1 and the digest output D_1 via HLSs, where $D_1 = digest(N_{\mathsf{C}}, \mathsf{C}, \mathsf{S}_1, CI)$. (b) Similarly, C sends S_2 *message 4* including S_2 and D_2 via HLSs, where $D_2 = digest(N_{\mathsf{C}}, \mathsf{C}, \mathsf{S}_2, CI)$. (c) *ep*: each slave node verifies the received digest value, and informs the user of the verification results.

Remark. The threat model and assumptions are

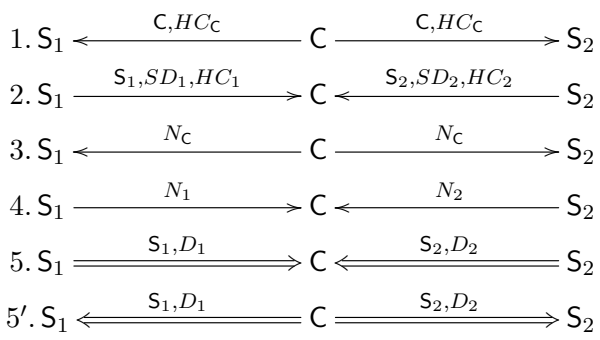
- HVLCs are NSB channels
- The user and all the nodes are trustworthy.
- group size is inputted via a NSB HCC.
- The notification/verification signal (e.g. LED alarm signal that should be read by human users) is transmitted over a NSB HCC.
- Button pushing procedure is a NSB HCC.

Whether we should use reveal phase step 4 or alternative step 4 is decided by the hardware. Reveal phase step 4 is used when the channel is established using LEDs in slave nodes (e.g. an Arduino node) and the camera in the coordinator (e.g. a mobile phone). Alternative reveal phase step 4 is used when the channel is established using LEDs in the coordinator (e.g. an Arduino node), and light sensors in slave nodes (e.g. an Arduino node).

HVSP-SD HVSP-SD is another HVSP that helps slave nodes to submit their sensor data SD_1 and SD_2 securely. The message flow is shown in Protocol 5.2. The details are explained below.

Commit phase step 1. (a) *bp*: the user inputs group size $gs = 3$ on C, and pushes a button on each node in order to initiate the protocol. (b) After this, C generates a long nonce N_{C} and a hash commitment $HC_{\mathsf{C}} = hash(N_{\mathsf{C}}, \mathsf{C})$. (c) C broadcasts *message 1* including its identity C, and hash commitment HC_{C} over wireless radio channels. (d) After message 1 has been received, each slave node ceases to accept any new commitment message.

Protocol 5.2 (HVSP-SD)**.**



Commit phase step 2. (a) After step 1.d, S_1 generates a long nonce N_1 and hash commitment $HC_1 = hash(N_1, \mathsf{S}_1)$. S_1 broadcasts *message 2* including its identity S_1, sensor data SD_1 and the commitment HC_1 over wireless radio channels. (b) After step 1.d, S_2 generates a long nonce N_2 and hash commitment $HC_2 = hash(N_2, \mathsf{S}_2)$. S_2 broadcasts *message 2* including its identity S_2, sensor data SD_2 and the commitment HC_2 over wireless radio channels. (c) *cp*: C counts the number of received different commitment messages. If the number is $gs - 1 = 2$, it ceases to accept any new commitment message, and then goes to step 3.

Reveal phase step 3. (a) *rp*: C broadcasts the opening message *message 3* including N_{C} over wireless radio channels. (b) Each slave node verifies HC_{C}. If the verification fails, the slave node aborts and informs the user. If the verification is successful, it goes to step 4.

Reveal phase step 4. (a) If the verification in step 3.b is successful, S_1 broadcasts the opening message *message 4* including N_1 over wireless radio channels; and S_2 broadcasts the opening message *message 4* including N_2 over wireless radio channels. (b) C verifies HC_1 and HC_2. If the verification fails, the node aborts and informs the user.

Reveal phase step 5. (a) S_1 computes $D_1 = digest(N_{\mathsf{C}} \oplus N_1, \mathsf{C}, \mathsf{S}_1, SD_1)$, and S_2 computes $D_2 = digest(N_{\mathsf{C}} \oplus N_2, \mathsf{C}, \mathsf{S}_2, SD_2)$. Each slave node transfers *message 5* including its digest value to C via HLCs. (b) *ep*: C verifies D_1 and D_2, and informs the user of the verification result.

Alternative reveal phase step 5. (a) C sends S_1 *message 4* including the target identity S_1 and the digest output D_1 via HLSs, where $D_1 = digest(N_{\mathsf{C}}, \mathsf{C}, \mathsf{S}_1, SD_1)$. (b) Similarly, C

sends S_2 *message 4* including S_2 and D_2 via HLSs, where $D_2 = digest(N_\mathsf{C}, \mathsf{C}, \mathsf{S}_2, SD_2)$. (c) *ep*: each slave node verifies the received digest value, and informs the user of the verification results.

Remark. The threat model and assumptions are

- HVLCs are NSB channels
- The user and all the nodes are trustworthy.
- group size is inputted via a NSB HCC.
- The notification/verification signal (e.g. LED alarm signal that should be read by human users) is transmitted over a NSB HCC.
- Button pushing procedure is a NSB HCC.

Whether we should use reveal phase step 5 or alternative step 5 is decided by the hardware. Reveal phase step 5 is used when the channel is established using LEDs in slave nodes (e.g. an Arduino node) and the camera in the coordinator (e.g. a mobile phone). Alternative reveal phase step 5 is used when the channel is established using LEDs in the coordinator (e.g. an Arduino node), and light sensors in slave nodes (e.g. an Arduino node).

Understanding example HVSPs Capabilities of HVLCs heavily influence the design of HVSPs. Topological characteristics decide whether one value or multiple values can be efficiently transmitted. HLSs are usually *high-speed one-to-multiple* channels, and the processing capabilities of the nodes are *high*; thus it is efficient to transmit several short digest values from one transmitter to several receivers in a short period.

Meanwhile, HLCs are usually *low-speed multiple-to-one* channels, and the processing capability of C is high, i.e. the camera can read multiple digests at once; thus we can compare multiple values efficiently in a short period.

For example, either HISP-CI or HISP-SD is a bundle of $gs-1$ parallel instances between the coordinator node and each slave node separately. In this case, we can avoid the need (present in HISPs) of transmitting data between slave nodes.

5.6 Chapter Summary

In this chapter, HVLCs and HVSPs are studied. HVLCs are one type of OOB channel. Using HVLCs, HVSPs can provide security in personal networks.

HVLCs are studied in personal networks. HVLCs are established based on VLCs. We have investigated and designed suitable VLC devices and communication techniques. Using these, we have established two kinds of channel: the LED-sensor channel and the LED-camera channel. Thanks to the visibility of VLCs, these channels can be easily controlled by human users. These VLCs controlled by human users are HVLCs. As we analyzed previously, many HVLCs are NSB OOB channels. Besides, we can see that networking properties heavily influence the design of protocols. Based on the features of HVLCs, parallel commitment schemes are used.

HVSPs are designed using HVLCs. The authenticity, integrity and freshness of messages transmitted via wireless radio channels can be guaranteed using HVLCs. Compared to HISPs, the main advantage of these protocols is that they require *less human work*; in addition, they *support small nodes* that have no displays. HVSPs are an improvement of HISPs in many cases.

Chapter 6

Use Cases: Human Visible Security Protocols

In this chapter, we study two HVSP[1] use cases. These use cases will show that HVSPs are useful in wearable personal networks.

The first use case is key establishment using HVSPs in wearable personal networks. These new protocols are designed based on an ECDH version of HISPs. Improvements: they can reduce the burden of human users by using HVLCs; in addition, they reduce the computation burden in slave nodes.

The second use case is a secure positioning system. We eliminate potential location spoofing attacks found in [106] using HVSPs. This system will be useful when VLCs become popular in the future.

Contents

[1] HV for HVLC, and SP for security protocol.

6.1 Key Establishment Protocols Using HVLCs

This section includes (1) key agreement protocols, (2) key transport protocols, and (3) related discussions.

6.1.1 Key Agreement Protocols

ECDH is a key agreement protocol in personal networks, introduced in Chapter 4; this protocol suffers from MITM attacks, and has been improved using HISPs. Here, instead of HISPs, we use HVSPs to eliminate MITM attacks. The new improved ECDH protocols are HVSP-K I and II.

HVSP-K I HVSP-K I is a key agreement protocol that can securely exchange ECDH public tokens between C and each S_i separately without worrying about MITM attacks. The message flow is shown in Protocol. 6.1. The details are explained below.

Protocol 6.1 (HVSP-K I)**.**

1. $\mathsf{C} \longrightarrow \forall\, \mathsf{S}_i : \mathsf{C}, EU_{\mathsf{C}}, HC_{\mathsf{C}}$
2. $\forall\, \mathsf{S}_i \longrightarrow \mathsf{C} : \mathsf{S}_i, EU_i$
3. $\mathsf{C} \longrightarrow \forall\, \mathsf{S}_i : N_{\mathsf{C}}$
4. $\forall\, \mathsf{S}_i \Longrightarrow_{HLC} \mathsf{C} : \mathsf{S}_i, D_i$

4'. $\mathsf{C} \Longrightarrow_{HLS} \forall\, \mathsf{S}_i : \mathsf{S}_i, D_i$

Commit phase step 1. (a) *bp*: the user pushes a buttons on C and each S_i in order to initiate the protocol. (b) The user inputs group size gs to C. (c) After gs has been received, C generates a long nonce N_{C} and a hash commitment $HC_{\mathsf{C}} = hash(N_{\mathsf{C}}, \mathsf{C})$. (d) C broadcasts *message 1* including its identity C, public token EU_{C}, and HC_{C} over wireless radio channels. (e) After message 1 has been received, S_i ceases to accept any new message 1.

Commit phase step 2. (a) After step 1.e, S_i broadcasts *message 2* including its identity S_i and public token EU_i over wireless radio channels. (b) After this, S_i flashes its LEDs. (c) If C has received $gs - 1$ message 2, it ceases to accept any new message 2, and it flashes

its LEDs. (d) *cp*: if all the nodes have flashed their LEDs, the user informs C to release the opening message i.e. message 3.

Reveal phase step 3. (a) *rp*: C broadcasts the opening message *message 3* including N_{C} over wireless radio channels. (b) Each S_i verifies HC_{C}. If the verification fails, S_i aborts and informs the user.

Reveal phase step 4. (a) S_i broadcasts *message 4* including its identity S_i and the digest output D_i via HLCs, where $D_i = digest(N_{\mathsf{C}}, \mathsf{C}, EU_{\mathsf{C}}, \mathsf{S}_i, EU_i)$. (b) *ep*: C verifies the received D_i, and it informs the user of the verification results.

Alternative reveal phase step 4'. (a) C sends S_i *message 4* including the target identity S_i and D_i via HLSs, where $D_i = digest(N_{\mathsf{C}}, \mathsf{C}, EU_{\mathsf{C}}, \mathsf{S}_i, EU_i)$. (b) *ep*: S_i verifies the received D_i, and it informs the user of the verification result.

HVSP-K II Alternatively, we can exchange ECDH public tokens using HVSP-K II. The message flow is shown in Protocol. 6.2. The details are explained below.

Protocol 6.2 (HVSP-K II)**.**

1. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : \mathsf{C}, EU_{\mathsf{C}}, HC_{\mathsf{C}}$
2. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} : \mathsf{S}_i, EU_i, HC_i$
3. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : N_{\mathsf{C}}$
4. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} : N_i$
5. $\forall \mathsf{S}_i \Longrightarrow_{HLC} \mathsf{C} : \mathsf{S}_i, D_i$

5'. $\mathsf{C} \Longrightarrow_{HLS} \forall \mathsf{S}_i : \mathsf{S}_i, D_i$

Commit phase step 1. (a) *bp*: the user inputs group size *gs* on C; and the user also pushes a button on each node in order to initiate the protocol. (b) After this, C generates a long nonce N_{C} and a hash commitment $HC_{\mathsf{C}} = hash(N_{\mathsf{C}}, \mathsf{C})$. (c) C broadcasts *message 1* including its identity C, public token EU_{C} and HC_{C} over wireless radio channels. (d) After message 1 has been received, each S_i ceases to accept any new message 1.

Commit phase step 2. (a) After step 1.d, S_i generates a long nonce N_i and hash commitment $HC_i = hash(N_i, \mathsf{S}_i)$. (b) S_i broadcasts *message 2* including its identity S_i, public token EU_i, and HC_i over wireless radio channels. (c) *cp*: C counts the number of re-

ceived different commitment messages. If the number is $gs-1$, it ceases to accept any new commitment message, and then goes to step 3.

Reveal phase step 3. (a) *rp*: C broadcasts the opening message *message 3* including N_{C} over wireless radio channels. (b) Each S_i verifies HC_{C}. If the verification fails, S_i aborts and informs the user; otherwise, it goes to step 4.

Reveal phase step 4. (a) S_i broadcasts the opening message *message 4* including N_i over wireless radio channels. (b) C verifies HC_i. If the verification fails, the node aborts and informs the user.

Reveal phase step 5. (a) S_i computes $D_i = digest(N_{\mathsf{C}} \oplus N_i, \mathsf{C}, EU_{\mathsf{C}}, \mathsf{S}_i, EU_i)$. Each S_i transfers *message 5* including its identity S_i and D_i to C via HLCs. (b) *ep*: C verifies D_i, and informs the user of the verification results.

Reveal phase step 5'. (a) C computes $D_i = digest(N_{\mathsf{C}} \oplus N_i, \mathsf{C}, EU_{\mathsf{C}}, \mathsf{S}_i, EU_i)$. C sequentially broadcasts *message 5* including the target identity S_i and D_i via HLSs. (b) *ep*: each S_i verifies D_i, and informs the user of the verification result.

Remark The threat model and assumptions for these two protocols are

- HVLCs are NSB channels
- The user and all the nodes are trustworthy.
- group size is inputted via a NSB HCC.
- The notification/verification signal (e.g. LED alarm signal that should be read by human users) is transmitted over a NSB HCC.
- Button pushing procedure is a NSB HCC.

Whether we should use reveal phase step or alternative step is decided by the hardware. Reveal phase step is used when the channel is established using LEDs in slave nodes (e.g. an Arduino node) and the camera in the coordinator (e.g. a mobile phone). Alternative reveal phase step is used when the channel is established using LEDs in the coordinator (e.g. an Arduino node), and light sensors in slave nodes (e.g. an Arduino node).

One example application scenario is as follows. A user wants to associate her mobile phone with several sensor nodes. She runs the protocol using the camera on her mobile phone and LEDs on sensor nodes. After the protocol is finished, keys are established between each pair of nodes.

6.1.2 Key Transport Protocols

Because HVSP-K I and II can only establish keys between C and each S_i separately, two new protocols are designed in order to support more flexible key schemes. These new protocols are HVSP-K III and IV. They are key transport protocols, which can transmit symmetric keys and other information from C to each S_i.

Key preparation C generates all keys, and the key selection algorithm is show in Procedure 6.1. For each node, the algorithm assigns keys based on the deployment adjacent matrix DAM that contains the topology information of the network.

Procedure 6.1 Key Selection

INPUT: Deployment adjacent matrix DAM, network size gs
OUTPUT: Random key matrix SHK and key assignment information matrix KS

1: **for** $i = 1$ to gs **do**
2: **for** $j = i + 1$ to gs **do**
3: **if** $DAM_{ij} > 0 \wedge KS_{ij} < 1$ **then**
4: Randomly generate one key shk;
5: $SHK_i = SHK_i \cup \{shk\}$;
6: $SHK_j = SHK_j \cup \{shk\}$;
7: $KS_{ij} = 1$;
8: $KS_{ji} = 1$;
9: **end if**
10: **end for**
11: **end for**
12: **return** SHK and KS

The assumption that we have DAM is reasonable. Firstly, using our protocols, the keys can be distributed on site; at this time, the usages and topologies of the small-scale networks are usually known. Secondly, the topologies are always simple, thus we can easily generate DAM.

Procedure 6.1 has the following advantages. It is better than the random key pool

scheme [40] for small-scale networks, since there are no isolated nodes or unused redundant keys. Additionally, this algorithm is an improvement of all pair-wise key schemes, since it reduces the number of useless keys stored in each node.

HVSP-K III HVSP-K III can transmit SHK and other information from C to each S_i. The message flow is shown in Protocol 6.3. The details are explained below.

Protocol 6.3 (HVSP-K III)**.**

1. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} : \mathsf{S}_i, PK_i$
2. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : \mathsf{C}, E_i, HC_{\mathsf{C}}$
3. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : N_{\mathsf{C}}$
4. $\forall \mathsf{S}_i \Longrightarrow_{HLC} \mathsf{C} : \mathsf{S}_i, D_i$

4'. $\mathsf{C} \Longrightarrow_{HLS} \forall \mathsf{S}_i : \mathsf{S}_i, D_i$

Commit phase step 1. (a) *bp*: the user inputs group size gs into C, and pushes a button on each node in order to initiate the protocol. (b) Each S_i broadcasts *message 1* including its identity S_i and public key PK_i over wireless radio channels. (c) C verifies that the number of received message 1 is $gs - 1$; if not, C aborts and sends a failure notification to the user, otherwise, it goes to step 2.

Commit phase step 2. (a) C generates a long nonce N_{C} and hash commitment $HC_{\mathsf{C}} = hash(N_{\mathsf{C}}, \mathsf{C})$. Suppose $encr()$ represents encryption, C computes $E_i = encr_{PK_i}(SHK_i)$. (b) C broadcasts its identity C, E_i and HC_{C} over wireless radio channels. (c) Each S_i that has finished step 1 and has received message 2 flashes its LEDs; and S_i ceases to accept new commitment message. (d) *cp*: if all the slave nodes have flashed their LEDs, the user informs C to release the opening message in step 3 by pushing a button.

Reveal phase step 3. (a) *rp*: C sends the opening message N_{C} to each S_i over wireless radio channels. (b) S_i verifies HC_{C}. If the verification fails, S_i aborts and informs the user.

Reveal phase step 4. (a) S_i computes $D_i = digest(N_{\mathsf{C}}, \mathsf{C}, \mathsf{S}_i, E_i, PK_i)$. (b) S_i transfers its identity S_i and D_i to C via HLCs. (c) *ep*: C verifies D_i, and informs the user of the verification results. If all the verifications are successful, S_i holds the authentic secret keys SHK_i.

Alternative reveal phase step 4′. (a) S_i computes $D_i = digest(N_\mathsf{C}, \mathsf{C}, \mathsf{S}_i, E_i, PK_i)$. (b) C transfers destination identity S_i and D_i to S_i via HLSs. (c) *ep*: each S_i verifies D_i, and informs the user of the verification result. If all the verifications are successful, S_i holds authentic the secret keys SHK_i.

HVSP-K IV HVSP-K IV is a symmetric version of HVSP-K III. It can transmit SHK and other information from C to each S_i. The message flow is shown in Protocol 6.4. The details are explained below.

Protocol 6.4 (HVSP-K IV)**.**

1. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} : \mathsf{S}_i, PK_i, HC_i$
2. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : \mathsf{C}, E_i, HC_\mathsf{C}$
3. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} : N_i$
4. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : N_\mathsf{C}$
5. $\forall \mathsf{S}_i \Longrightarrow_{HLC} \mathsf{C} : \mathsf{S}_i, D_i$

5'. $\mathsf{C} \Longrightarrow_{HLS} \forall \mathsf{S}_i : \mathsf{S}_i, D_i$

Commit phase step 1. (a) *bp*: the user inputs group size gs into C, and pushes a button on each node in order to initiate the protocol. (b) Each S_i generates a long nonce N_i and $HC_i = hash(N_i, \mathsf{S}_i)$. (c) S_i broadcasts *message 1* including its identity S_i, public key PK_i, and HC_i via wireless radio channels. (d) C verifies that the number of different message 1 is $gs - 1$; if not, C aborts and sends a failure notification to the user. If the verification is successful, C goes to step 2.

Commit phase step 2. (a) C generates a long nonce N_C and hash commitment $HC_\mathsf{C} = hash(N_\mathsf{C}, \mathsf{C})$. In addition, C computes $E_i = encr_{PK_i}(SHK_i)$. (b) C broadcasts *message 2* including its identity C, E_i and HC_C over wireless radio channels. (c) *cp*: each S_i that has sent message 1 and has received message 2 ceases to accept new commitment message; and S_i goes to step 3.

Reveal phase step 3. (a) *rp*: S_i sends the opening message N_i to C over wireless radio channels. (b) C verifies HC_i. If the verification fails, C aborts the protocol and informs the user. If the verification is successful, C goes to step 4.

Reveal phase step 4. (a) C sends the opening message N_{C} to each S_i over wireless radio channels. (b) S_i verifies HC_{C}. If the verification fails, S_i aborts the protocol and informs the user.

Reveal phase step 5. (a) S_i computes $D_i = digest(N_{\mathsf{C}} \oplus N_i, \mathsf{C}, \mathsf{S}_i, E_i, PK_i)$. (b) S_i transfers its identity S_i and D_i to C via HLCs. (b) *ep*: C verifies D_i, and informs the user of the verification results. If all the verifications are successful, S_i holds the authentic secret keys SHK_i.

Alternative reveal phase step $5'$. (a) C computes $D_i = digest(N_{\mathsf{C}} \oplus N_i, \mathsf{C}, \mathsf{S}_i, E_i, PK_i)$. (b) C transfers destination identity S_i and D_i to S_i via HLSs. (b) *ep*: S_i verifies D_i, and informs the user of the verification result. If all the verifications are successful, S_i holds the authentic secret keys SHK_i.

Remark The threat model and assumptions for these two protocols are

- HVLCs are NSB channels
- The user and all the nodes are trustworthy.
- group size is inputted via a NSB HCC.
- The notification/verification signal (e.g. LED alarm signal that should be read by human users) is transmitted over a NSB HCC.
- Button pushing procedure is a NSB HCC.

Whether we should use reveal phase step or alternative step is decided by the hardware. Reveal phase step is used when the channel is established using LEDs in slave nodes (e.g. an Arduino node) and the camera in the coordinator (e.g. a mobile phone). Alternative reveal phase step is used when the channel is established using LEDs in the coordinator (e.g. an Arduino node), and light sensors in slave nodes (e.g. an Arduino node).

One example application scenario is as follows. A user wants to associate her mobile phone with several sensor nodes. She runs the protocol using the camera on her mobile phone and LEDs on sensor nodes. After the protocol is finished, keys are established among

nodes. Note that some slave nodes may not have shared keys, which is different from HVSP-K I and II.

6.1.3 Discussion

These key establishment protocols using HVLCs are parallel protocol. Parallel means that there are no communications among S_i. However, it is still possible to establish shared keys among these nodes; because C can generate keys for these nodes, and transport these keys to them.

The human burden is reduced compared to HISP-K I and II. The comparison is shown in Table 6.1. In HVSP-K I, II, III, and IV, human users are not required to compare the digests. Certainly, in all these protocols, human users still need to control group sizes and to prevent physical attacks.

Table 6.1: Human burden

	HVSP-K I, II, III, and IV	HISP-K I and II
Digest comparison	No	$16 \times gs$ bits
Group size control	Require	Require
Prevent physical attacks	Require	Require

Besides, their computation consumption is roughly compared in Table 6.2. AsyC is the number of asymmetric key cryptography computations in the coordinator; AsyS is the total number of asymmetric key cryptography computations in slave nodes. The computation consumption of these protocols is optimized for slave nodes, compared to the ECDH version of HISPs. The total number of asymmetric cryptographic computations in slave nodes are shown in Fig. 6.1. The x axis is the group size, and the y axis is AsyS. The computation consumption of the coordinator is neglected, since it always directly connects to computers. From Fig. 6.1, we can see that HVSP-K I, II, III and IV are much better than HISP-K I and II.

Finally, the security analysis of HVSP-K I, II, III and IV is similar to the analysis in Chapter 3. The authenticity, integrity and freshness of the ECDH public tokens and the encrypted secret keys $encr_{PK_i}(SHK_i)$ can be guaranteed. As long as the cryptographic algorithms hold the desired security properties, these protocols are good authentication

Table 6.2: Efficiency Comparison

	AsyC	AsyS
HVSP-K I	$gs-1$	$gs-1$
HVSP-K II	$gs-1$	$gs-1$
HVSP-K III	$gs-1$	$gs-1$
HVSP-K IV	$gs-1$	$gs-1$

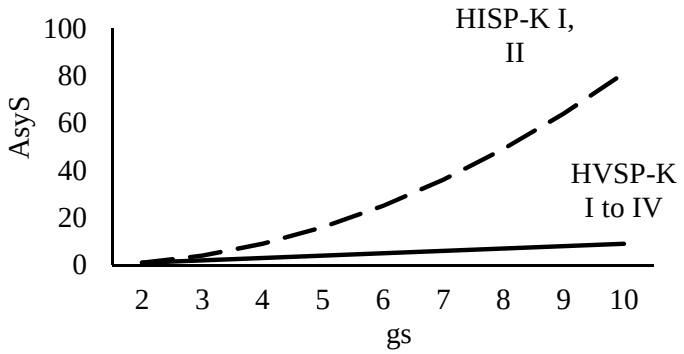


Figure 6.1: Computation efficiency comparison

and key distribution protocols.

6.2 Secure Location Service

The section is organized as follows: (1) an explanation of location services in personal networks; (2) a review of attacks on the location services; and (3) the countermeasures using HVSPs.

6.2.1 Location Services in Personal Network Applications

Many personal network applications involve location services. Especially in medical applications and context-aware computing, positioning systems are always one of the important sub-systems. They are discussed below.

In medical applications, location services are always integrated into the system. For example, in AWARENESS [116], location information regarding the patients and the nearest healthcare professionals are collected; in CodeBlue [74], localization is used in order to track patients in a disaster response scenario; and AID-N [41] also uses positioning systems for a medical emergency response.

Figure 6.2: Ambiguous location of Jerry

Additionally, in context-aware computing, location information is always used as one important context. One example is PrivacyIoT [51, 52]. This provides privacy preserved location service using context-aware access control, which relies heavily on the users' locations. One example is shown below.

Example. Tom wants to know Jerry's location. Tom's pre-defined role is colleague. When Jerry's context is

$$\left\{ \begin{array}{c} \langle \text{Time, 9:00AM} \rangle, \\ \langle \text{Location, University} \rangle, \\ \langle \text{Event, On duty} \rangle \end{array} \right\}.$$

Tom is assigned to the context aware role "colleague@on duty"; with this, Tom can get the exact location of Jerry. However, when Jerry's context is

$$\left\{ \begin{array}{c} \langle \text{Time, 9:00PM} \rangle, \\ \langle \text{Location, Home} \rangle, \\ \langle \text{Event, Off duty} \rangle \end{array} \right\}.$$

Tom's context aware role becomes "colleague@off duty"; and Tom can only know an ambiguous location, which is shown in Fig. 6.2. □

6.2.2 Attacks Against Location Services

Many location services use WLAN positioning systems as a substitution for and a complement to the Global Positioning System. One typical system is the Wi-Fi positioning system (WPS) from Skyhook [6]. There are three parties: localized node (LN) that issues the requests of location; WLAN access points (AP) that typically broadcast service announcement beacons from fixed known locations; and the operator (Skyhook) that has a

location look up table (LLT). The Skyhook localization process is shown in Fig. 6.3, and it is explained in Protocol 6.5.

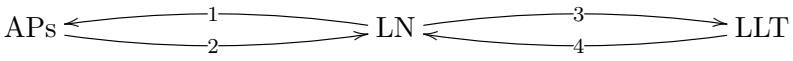


Figure 6.3: Skyhook localization process

Protocol 6.5 (Skyhook localization process)**.**

1. LN broadcasts a probe request frame.
2. APs reply with a response beacon frame.
3. LN queries the LLT server.
4. LLT server returns location data of the observed APs.
5. LN computes its location.

The authors of [106] have analyzed the security of the Skyhook WPS, and demonstrated that the positioning system is vulnerable to location spoofing. Location spoofing attacks mean that the attacker can arbitrarily change the localization results at the victim device by signal insertion, replay and jamming.

The first location spoofing attack is an AP impersonation attack. Suppose LN is not in the range of legitimate APs, this attack is shown in Fig. 6.4, and is explained in Attack 6.1. By impersonating APs as described, the localization process on the iPod returned a location in New York City, while the device was physically located in Europe.

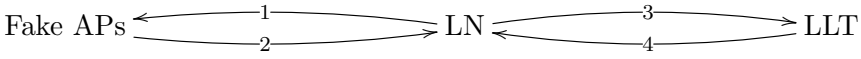


Figure 6.4: AP impersonation attack

The second location spoofing attack is named an AP replacement attack, shown in Fig. 6.5. Suppose LN is in the range of legitimate APs and uses only WLAN-based localization. The attacker can jam the signal of legitimate APs, and then attack the protocol. The attack procedure is similar to the AP impersonation attack.

The third attack is an attack against hybrid WLAN/GSM localization systems. Since

Attack 6.1 (AP impersonation attack)**.**

1. LN broadcasts a probe request frame.
2. Fake APs reply with a response beacon frame using remote MAC addresses.
3. LN queries the LLT server using these MAC addresses.
4. LLT server returns location data about the observed APs.
5. LN computes its location.

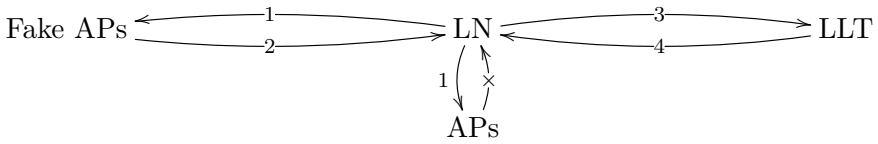


Figure 6.5: AP replacement attack

GSM localization is significantly less accurate than WPS localization, the hybrid system will use WPS localization whenever it is available. Thus, similar attacks mentioned previously can also compromise the hybrid system.

6.2.3 Location Service Protocols

Location service protocols using HVLCs are proposed here. If VLCs become popular in the next few years, it is reasonable to assume that APs and ceiling LEDs will be securely connected. In this case, these proposed protocols can be useful.

HVSP-L[2] I is the first protocol. The protocol is described in Protocol 6.6. Using this protocol, the location spoofing attacks can be eliminated. If an attacker uses fake WLAN APs and changes their MAC addresses, the attack can be found in the third step. Note that the threat model and assumptions are

- HLSs are NSB channels.
- APs and ceiling LEDs will be securely connected.

[2]L for localization

Protocol 6.6 (HVSP-L I)**.**

1. LN broadcasts a probe request frame.
2. APs reply with response beacon frames.
3. LN and ceiling LEDs exchange the hash output of the request frame and the response frames via HLSs.
4. LN queries the LLT server.
5. LLT server returns location data about the observed APs.
6. LN computes its location.

Alternatively, we can directly use HVLCs and ceiling LEDs instead of WLAN and APs in the localization process. The protocol steps are listed in Protocol 6.7. When HVLCs are NSB channels, modifying or blocking messages exchanged in the first and second steps can be eliminated. In addition, the localization results can be more accurate than the WLAN-based approaches. Note that the threat model and assumptions are

- HLSs are NSB channels.
- Ceiling LEDs are trustworthy.
- Ceiling HLS receiver is trustworthy.

Protocol 6.7 (HVSP-L II)**.**

1. LN broadcasts a probe request frame to the ceiling HLS receiver.
2. LEDs reply with response beacon frames using HLSs.
3. LN queries the LLT server.
4. LLT server returns location data about the observed LEDs.
5. LN computes its location.

6.3 Chapter Summary

This chapter contains two HVSP use cases. These use cases show that HBSPs can be building blocks of future security mechanisms in personal networks.

The first use case is key establishment protocols using HVLCs. Firstly, instead of HISPs, we use HVSPs in order to eliminate MITM attacks against ECDH protocols. These protocols can establish keys between a coordinator and each slave node separately. When connections among slave nodes are not necessary, these protocols are a good solution. Secondly, in order to support flexible key schemes, we combine HVSPs with key pre-distribution schemes. These protocols do not require an absolute secure phase to distribute keys. HVLC-based key establishment protocols have the following advantages. Firstly, they can reduce the burden of human users, i.e. human users are not required to compare strings/digits/a series of flash patterns. Secondly, the computation consumption is optimized for slave nodes. Since slave nodes are generally more liable to energy- and computation-constraint than a coordinator, this optimization is meaningful.

The second use case is a secure positioning system. Location services have become popular in current personal network applications, especially medical applications and context-aware applications. For example, AWARENESS [116], CodeBlue [74] and AID-N [41] mentioned in Chapter 2 all provide location services. Recently, authors of [106] have demonstrated that positioning systems can be vulnerable to location spoofing. Location spoofing attacks mean that the attacker can arbitrarily change the localization results at victim devices by signal insertion, replay and jamming. Thus, we have designed two HVSPs, which can eliminate the location spoofing attacks. If VLC becomes popular in the next few years, it is reasonable to assume that access points and ceiling LEDs will be securely connected. In this case, our protocols will be a better choice in comparison with current solutions.

In summary, HVLCs are promising techniques. They have various desired security properties, and their realizations are simple and low-cost. They could be one of the fundamental technologies of future security mechanisms.

Chapter 7

Intra-Body Communications and Security Protocols

HBSPs[1] are protocols designed for providing security in wearable and implanted personal networks using HBCs[2]. The concept design is as follows. In personal networks, wireless radio channels are used for data communications, and properly selected HBCs can protect the data; or properly selected HBCs can be used to directly transmit data securely.

Comparing to HISPs and HVSPs, HBSPs can provide security to implants; this is their main advantage. In addition, HBSPs can reduce the burden of human users and support smaller nodes, which are an improvement over HISPs. Finally, HBSPs are also flexible: they do not require pre-distributed secrets.

Contents

[1] H for human, B for body, S for security, P for protocol
[2] H for human, B for body, and C for channel

Table 7.1: Comparison of IBCs [118]

	Amplitude	Data rate (bps)
Fukumoto	21V	0.1 k
Zimmerman	30V	2.4 k
Reynolds	10V	9.6 k
Partridge	22V	38.4 k
Hachisuka	1V	9.6 k
NTT/Docomo	25V	10 m
Handa	20μA	0.9 k
Oberle	4mA	4.8 k

7.1 IBCs

This section includes (1) background of IBCs, (2) channel establishment technologies, and (3) some experiment results.

7.1.1 Background

Intra-body channels (IBCs) are channels that use the human body as transmission media, and their conceptual design is shown in Fig. 7.1. Loosely speaking, each node is associated with an IBC transmitter; it sends electric signals via human tissue. Meanwhile, each node is also associated with one receiver; it can receive signals from human tissue.

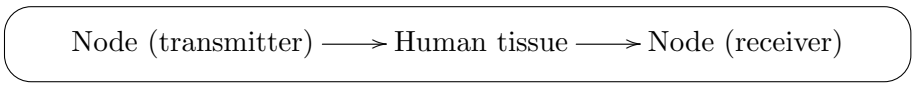


Figure 7.1: IBC

In [117], the first successful IBC system is reported, and many interesting works have been published since then. In Table 7.1, state-of-art IBCs are summarized. The available data rates are from 0.1 kbps to 10 mbps. They are fast enough to transfer digest outputs, hash outputs, and even cryptographic keys. In addition, the tested amplitudes in volts are from 1 V to 30 V. Since the output voltages from the nodes are typically between 3 V and 7 V, we have reason to believe that IBCs can be established using typical nodes.

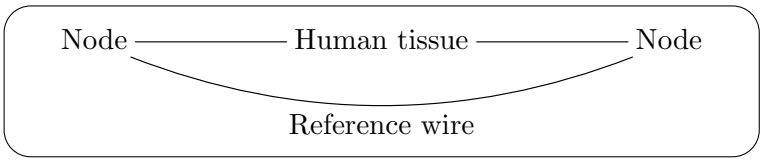


Figure 7.2: Direct transmission

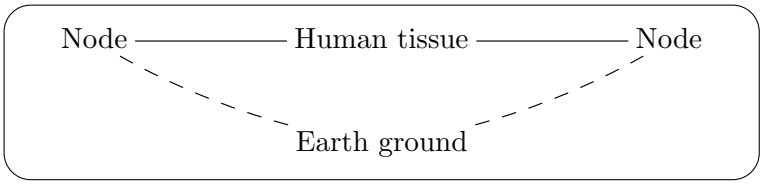


Figure 7.3: Capacitive coupling

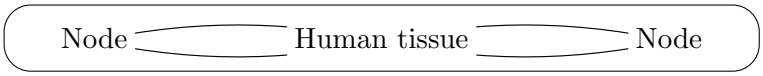


Figure 7.4: Galvanic coupling

7.1.2 Channel Establishment

Devices design In principle, there are three types of IBC: direct transmission, capacitive coupling, and galvanic coupling. Direct transmission is shown in Fig. 7.2: the electrodes of nodes are directly attached to the skin; in addition, transmitter and receiver share a common reference connected by a wire. Capacitive coupling is shown in Fig. 7.3: each node has a signal electrode that is directly attached to the skin, and the two nodes share an "earth ground" to which they are both capacitively coupled; however, capacitive coupling is sensitive to the environment. Galvanic coupling is shown in Fig. 7.4: each node has two electrodes that are attached to the skin; major current flow occurs between the two electrodes of transmitter; there is also a small current flow between the two electrodes of receiver; and the receiver gets signals by measuring the differential signals between its two electrodes.

Communication procedure IBC establishment procedure is shown in Fig. 7.5. The procedure includes the following steps.

Step 1. Data is encoded into message frames shown in Fig. 7.6. Preamble (1010) is used for synchronization; and delimiter (11) denotes the end of the preamble. Sender ID is

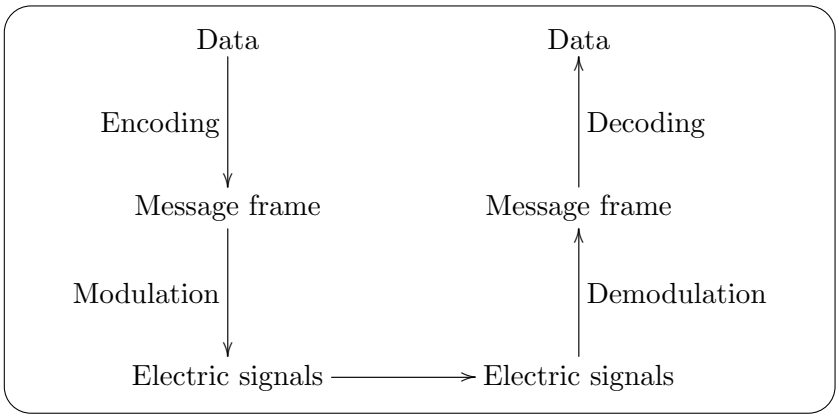


Figure 7.5: IBC channel establishment procedure

Preamble	Delimiter	Sender ID	data

Figure 7.6: Message Frame

a 5-bit binary code. Data length depends on the content length; for example, the content could be RSA public key, hash output, ECDH public token, or digest output.

Step 2. Each bit of the frame is modulated to electric signals by the modulator. We use on-off keying modulation; simply speaking, 1 in the message frame is modulated as High, and 0 is modulated as Low.

Step 3. Electric signals are transmitted via human tissue; and the receiver measures signals in a high sample rate. Note that the voltages of the transmitter ports are lower than 7 V. It is safe for the human body in most cases; however, safety issues must be carefully examined before deployment. Readers interested in the electrical models of IBCs can find technical details in [118].

Step 4. Received electric signals are demodulated to message frames by the demodulator. This process is as follows: the receiver firstly computes a voltage threshold; and then the received signal is 1 if the voltage measured by the receiver is higher than the threshold, otherwise the signal is 0.

Step 5. Message frames are decoded to data. Simply speaking, data is extracted from the message frames, and preambles and delimiters are discarded.

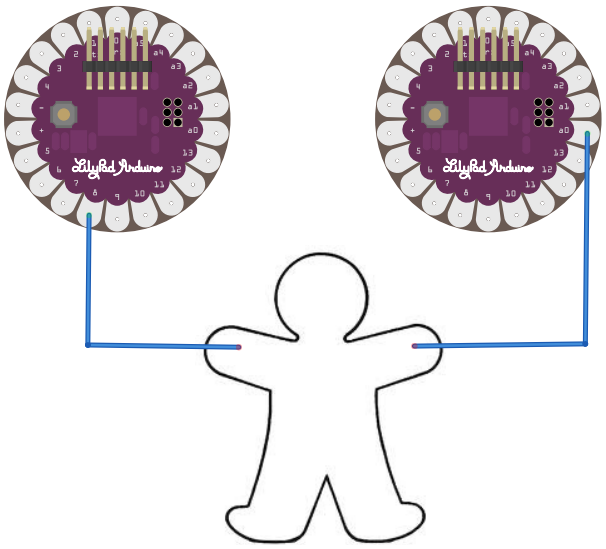


Figure 7.7: Example of IBC

7.1.3 Experiments

None of the IBC equipment was found commercially available, except for BodyCom[3] that does not support general sensor network platforms currently, so we built our own experimental implementation. The experiment system is a capacitive coupling system; it consists of two nodes powered by USB[4] sockets, shown in Fig. 7.7. The transmitter is the digital port D8 on the left node; this port can output two logical signals 1 (High) and 0 (Low) by changing its voltage. The receiver is the analogue port A0 on the right node; it reads input voltages, and maps the voltages into integer values between 0 and 1023. Human users can either directly touch two ports and connect these two nodes, or they can use wires, especially conductive thread[5] or fabric[6].

The system specifications are introduced as follows. The voltage of port D8 is between 0 V to 3.3 V, and thus is safe for a normal person (electrical properties of their body are normal). The raw output bit rate is around 2 kbps. The maximum sample rate of the analogue port is around 10 kHz, thus the maximum bit rate should be less than 5kbps.

[3] BodyCom: http://www.microchip.com/pagehandler/en-us/technology/embeddedsecurity/technology/bodycom.html, access time: 2013-11-01

[4] universal serial bus

[5] An example of conductive thread can be found at: https://www.sparkfun.com/products/10867, access time: 2013-08-06

[6] An example of conductive fabric can be found at: https://www.sparkfun.com/products/10070, access time: 2013-08-06

The first experiment was the transmission time of a 200-bit message. Port D8 was attached to the left hand, and port A0 was attached to the right hand. We tested the message transmission 10 times. All messages were correctly received within 1 second.

In the second experiment, we tested the following scenarios: port D8 was attached to the left hand, and port A0 was attached to the left arm; port D8 was attached to the left hand, and port A0 was attached to the left leg; port D8 was attached to the left hand, and port A0 was attached to the right arm; port D8 was attached to the left hand, and port A0 was attached to the right leg. In every scenario, the 200-bit message was successfully transmitted.

7.2 HBCs

This section includes (1) channel overview, (2) security properties, and (3) networking properties.

7.2.1 Channel Overview

HBCs are defined in Definition 7.1 below. The human user can choose an occasion to run the protocol when the HBC is not being interfered with. This is because the channel is established through the body, and the body's owner can isolate her/himself from sources of electrical interference. For example, the user should avoid touching someone else, and preferably only run the protocol in a safe environment.

Definition 7.1 (HBC). HBCs are IBCs controlled by human users. We assume that the human should only use this channel where no interference is likely, and is able to perceive any unexpected contact and cancel a protocol run. In addition, human users can determine (1) which node will be involved in a certain communication procedure; (2) when the node will be involved in a certain communication procedure; and (3) when the information will be transmitted.

Nodes are worn by human users, or implanted inside the human body; thus users have the capability of controlling (1) to (3) in the definition. In addition, HBCs can use the

human body as a communication channel, as long as users avoid being touched by others during the communications, and interference can be avoided.

7.2.2 Security Properties

Advantages HBCs offer several security advantages over existing wireless radio technologies. We list these as follows:

- Authenticity. HBC signals are transmitted through a user's body. As long as users have not been attached to any illegal signal source; for example if users are not touched by others during the communications, the authenticity of HBC signals can be guaranteed.
- Integrity. For similar reasons, the integrity of HBC signals can be guaranteed.
- Conditional confidentiality. The reported signal leakage distance is around 1 metre [129]. Thus, as long as users establish HBCs in trustworthy places, i.e. users believe there to be no malicious signal detectors nearby, HBCs can be used as conditional confidential channels. Generally speaking, examples of trustworthy places include home and office; banks or ATMs are not good examples, because attackers can easily set up malicious detectors in these places.

NSB HBCs With advantages mentioned above, blocking or spoofing messages over these channels can be eliminated. These HBCs are defined in Definition 7.2.

Definition 7.2 (NSB HBC). The NSB HBC is a HBC, and should be modelled as a NSB channel from a security standpoint.

Threat model and assumptions Threat model. HBCs that will be used in our protocols are assumed to be NSB channels. In other words, attackers can overhear messages over these channels, however they cannot block, modify, fake, replay, or delay messages. This threat model is based on assumptions elaborated below.

Assumption 1: users are trustworthy. It means that users will not attack nodes or protocols. Also, they will follow the instructions in protocols.

Assumption 2: nodes are trustworthy. In other words, there are no malicious programs that will change messages transmitted or received

Assumption 3: attacks that try to block, modify, fake, replay, or delay messages over HBCs can be found and prevented by users.

This assumption is reasonable. During the communication procedure, in order to block or spoof HBC messages, the attacker generally has the following strategies.

1. A sends fake or interference signals by attaching malicious signal sources to a user's skin. Such an attack can be perceived and prevented by a normal person (with normal perceptions, can feel that they are touched by others).

2. A uses a fake node instead of a legal node. This is impossible since the nodes are controlled by users.

However, it is unreasonable to assume that users can prevent an attacker A from communicating with their implants over IBCs by, for example, handshaking and hugging. Similar attacks can be easily perceived and prevented in wearable networks (largely thanks to buttons). Thus, for these implants, additional access control mechanisms are required in order to prevent such illegal communications.

Assumption 4: attackers can overhear messages over HBCs in some cases (Conditional confidential channel).

The reported signal leakage distance is around 1 metre [129]. Thus, it is possible to overhear messages over HBCs without being noticed by users.

User expectations (1) When users are running protocols, especially when messages are transmitting over HBCs among nodes, users should avoid being touching. (2) Users should update operating systems or even use anti-virus software in order to make sure that their nodes are trustworthy. (3) Users should not allow others to use their nodes unless a) they are trustworthy, and b) this is necessary.

7.2.3 Networking Properties

The networking characteristics of HBCs are one important factor of protocol design. Their topology properties and speed largely determine their usages. Thus, we analyze their networking characteristics as follows.

The transmission rate of HBCs can be relatively high. Transmission speeds of HBCs are determined by IBCs. According to our experiments, the speed can achieve 0.2k bps; and in Table 7.1, we can see that the speed can achieve 10 mbps. The speed is fast enough to transmit public keys, especially hash outputs and digest outputs.

HBCs are naturally one-to-multiple channels. Multiple receivers attached to the same human body can receive electric signals from one transmitter at the same time. Certainly, using multiplexing techniques, for example time-division multiplexing and frequency-division multiplexing, HBCs can become multiple-to-multiple channels.

7.3 Security Protocols based on HBCs

This section includes: (1) HBSPs, (2) example HBSPs for wearable nodes, (3) HBSPs for implants, (4) example HBSPs for implants, and (5) related discussions.

7.3.1 HBSPs

HBSPs are distinguished from other multi-channel security protocols by the properties of HBCs. The definition is given in Definition 7.3.

Definition 7.3 (HBC-based security protocol (HBSP)). A HBSP is a protocol that provides security based on the properties of HBCs.

Since HBCs support high-speed communications, long messages (for example public keys) can be directly transmitted. HBCs can also be used for verifying messages exchanged in insecure wireless radio channels. In either case, security requirements, for example authenticity, integrity and freshness, can be guaranteed thanks to HBCs.

7.3.2 Examples of HBSPs for Wearable Nodes

HBSP-CI HBSP-CI is one example HBSP. Suppose there is a personal network that consists of one mobile phone C, and two slave nodes S_1 and S_2. HBC-CI can help C to deliver instructions to CI securely. The message flow is shown in Protocol. 7.1. The details are explained below.

Protocol 7.1 (HBSP-CI)**.**

$$1.\ \mathsf{S}_1 \xleftarrow{\mathsf{C},CI} \mathsf{C} \xrightarrow{\mathsf{C},CI} \mathsf{S}_2$$
$$2.\ \mathsf{S}_1 \xRightarrow{\mathsf{S}_1,H_1} \mathsf{C} \xLeftarrow{\mathsf{S}_2,H_2} \mathsf{S}_2$$

Step 1. (a) The user pushes the button on each node to initiate the protocol. (b) The user inputs group size $gs = 3$ into C. (c)After gs has been received, C broadcasts its identity C and instructions CI over wireless radio channels.

Step 2. (a) S_1 sends its identity S_1 and $H_1 = hash(\mathsf{C}, \mathsf{S}_1, CI)$ to C over HBCs. (b) S_2 sends its identity S_2 and $H_2 = hash(\mathsf{C}, \mathsf{S}_2, CI)$ to C over HBCs. (c) C verifies H_1, H_2, and the group size; if verification fails, C gives an alarm.

HBSP-SD HBSP-SD is another example of HBSP. It can help slave nodes to submit their sensor data SD_1 and SD_2 securely. The message flow is shown in Protocol. 7.2. The details are explained below.

Protocol 7.2 (HBSP-SD)**.**

$$1.\ \mathsf{S}_1 \xLeftarrow{\mathsf{C}} \mathsf{C} \xRightarrow{\mathsf{C}} \mathsf{S}_2$$
$$2.\ \mathsf{S}_1 \xRightarrow{\mathsf{S}_1,SD_1} \mathsf{C} \xLeftarrow{\mathsf{S}_2,SD_2} \mathsf{S}_2$$

Step 1. (a) The user pushes the button on each node to initiate the protocol. (b) The user inputs group size $gs = 3$ into C. (c)After gs has been received, C broadcasts its identity C over HBCs.

Step 2. (a) After message 1 has been received, slave node $\mathsf{S}_1/\mathsf{S}_2$ replies with its identity $\mathsf{S}_1/\mathsf{S}_2$, and sensor data SD_1/SD_2 over HBCs. (b) C counts the received message 2; if the number is $gs - 1 = 2$, C informs the user with a signal of success, otherwise C informs the

user with a signal of failure.

Remark The threat model and assumptions are

- HBCs are NSB channels
- The user and all the nodes are trustworthy.
- group size is inputted via a NSB HCC.
- The notification/verification signal (e.g. LED alarm signal that should be read by human users) is transmitted over a NSB HCC.
- Button pushing procedure is a NSB HCC.

These two protocols are used in wearable networks. The nodes are typically Arduino Lilipad or even smaller. They can be much smaller than nodes used in previous chapters. However, they must have ports that can transmit and receive electrical signals.

7.3.3 HBSPs for Implants

HBSPs can be used for implants: implants are inside the human body, and HBSPs can communicate with the implants using HBCs. These protocols will become more and more important when personal networks that include implants are increasingly used in the real world. Note that, although HISPs or HVSPs can also be used in some specific cases, for example retinal implants, they are not suitable for implants in general.

HBSPs for implants are different to previous multi-channel security protocols. First of all, it is difficult to establish human-controlled channels between human users and slave nodes. This is because human-controlled channels, for example button pushing and LED flashing, which were important in previous security protocols, can rarely be realized for implants. Note that coordinator nodes are assumed to be computers or mobile phones, so interactions between them and human users will not be restricted.

Additionally, it is difficult for users to prevent an attacker A from communicating with their implants over IBCs by, for example, handshaking and hugging. Thus, HBSPs for implants need additional access control mechanisms. These access control mechanisms

could be PIN codes that are transmitted via HBCs in order to initiate communication. Alternatively, HBCs can only be enabled by some faithful devices or channels such as specific pre-configured devices, physical pressure, shaking, or (ultra)sound. The entrance security control system is a good analogy.

Besides, implanted nodes are assumed to be extremely computation-constrained, thus it is better to use computation-efficient algorithms. This does not mean that asymmetric cryptography should never be used; it depends on specific use cases. Moreover, denial-of-service attacks such as battery depletion should be carefully prevented.

Finally, security for implants can be highly sensitive to context. For example, in an emergency situation, the implants might be required to relax their security mechanisms so that first aid can be given quickly. However, strong security is required to prevent privacy leakage and other attacks in everyday life.

7.3.4 Examples of HBSP for Implants

Assume that there is a mobile phone M, an implanted node S, a wearable coordinator node C, and a secondary coordinator node C'. In order to achieve a balance between security and safety, we adopt fine-grained mechanisms: different users are assigned different roles; and different roles are associated with different permissions.

In Table 7.2, we show the roles and permissions in this example. Owner, emergency contact and health authority are three roles; and data access, device configuration, and authorization are three permissions. Owner has full permissions: it can access data stored in the implants, configure the implants, and authorize permissions to others; this role is usually assigned to the implants owner. Emergency contact can authorize permissions to others; this role is usually assigned to the party that will be contacted in an emergency situation. The health authority can access the data to monitor the owner's health status, and can configure the implants in order to treat the owner; this role is usually assigned to the party that provides medical services.

Initialization phase In this phase, S has not yet been implanted. Thus, we can bootstrap security between S and other devices using our previous protocols easily. In this example,

Table 7.2: Roles and permissions

Roles	Data access	Device configuration	Authorization
Owner	√	√	√
Emergency contact			√
Health authority	√	√	

we want to achieve the following goals:

- establish shared key SHK between S and C;
- establish shared key SHK' between S and C';
- the owner sets a password PWD in S;
- the emergency contact sets a password PWD' in S.

Note that the threat model and assumptions are

- The HBCs are confidential and NSB channels.
- The user, S, C and C' are trustworthy.

Step 1. Since the initialization usually happens in a trusted place, we can directly use HBCs as confidential and NSB channels. The SHK and SHK' establishment process is shown in Protocol 7.3. Firstly, the owner switches on S and C in order to initiate the protocol; and S sends its identity to C and C' via HBCs. Secondly, C tells S its identity and SHK via HBCs. Thirdly, C' tells S its identity and SHK' via HBCs. More initialization protocols will be discussed in the next chapter.

Protocol 7.3 (Initialization protocol)**.**

1.1 $\mathsf{S} \Longrightarrow \mathsf{C} \wedge \mathsf{C}' : \mathsf{S}$

1.2 $\mathsf{C} \Longrightarrow \mathsf{S} : \mathsf{C}, SHK$

1.3 $\mathsf{C}' \Longrightarrow \mathsf{S} : \mathsf{C}', SHK'$

Step 2. C is given to the owner; and a password PWD is set using C by the owner. In addition, C' is given to the emergency contact; and a password PWD' is set using C' by the emergency contact. C, C', PWD, and PWD' will be used for authorization after S is implanted.

Authorization Assume S is implanted from now on. In addition, suppose there is a node M; the first example of M is the owner's mobile phone, which is used to monitor his/her health status; the second example of M is first-aid equipment belonging to the health authority, which is used to treat the owner in an emergency situation. Now, suppose M wants to access S, and C is worn by the owner. M can run Protocol 7.4 to get a session key and a session timer. The steps are explained below.

Protocol 7.4 (Authorization protocol)**.**

1. $\mathsf{M} \longrightarrow \mathsf{C} : \mathsf{M}, \mathsf{S}$
2. $\mathsf{C} \longrightarrow \mathsf{M} : \{\mathsf{S}, SHK_{\mathsf{MS}}, \text{Timer}\}_{SHK_{\mathsf{MC}}}$
3. $\mathsf{M} \Longrightarrow \mathsf{C} : H$
4. $\mathsf{C} \Longrightarrow \mathsf{S} : msg_4, \mathsf{S}, \{\mathsf{M}, SHK_{\mathsf{MS}}, \text{Timer}\}_{SHK}$

Step 1. (a) The user pushes buttons on C and M in order to initiate the protocol. (b) After the button pushing, M broadcasts its identity M and the objective identity S over wireless radio channels.

Step 2. After message 1 has been received, C sends out message 2. SHK_{MS}, which is generated by C, is a shared key between M and S; a timer indicates the beginning and expiration time of this session; and SHK_{MC} is a shared key between M and C, which can be easily established using our previous multi-channel protocols.

Step 3. (a) After message 2 has been received, M replies with H via HBCs, where

$$H = hash(\mathsf{M}, \mathsf{S}, \mathsf{C}, \{\mathsf{S}, SHK_{\mathsf{MS}}, \text{Timer}\}_{SHK_{\mathsf{MC}}}).$$

(b) C verifies H. If the verification succeeds, it goes to step 4.

Step 4. (a) C sends out message 5 via HBCs. SHK is the shared key between S and C. msg_4 is a message label. (b) S will only be activated when it sees the label signal msg_4. (c) S receives and decrypts message 5 using SHK. If the decryption fails (i.e. the decryption result is a garbage), it stops from receiving any messages for a while (this period should be configured based on the requirement of real-world applications); this can also help S to prevent DoS attacks (refers to battery depletion attacks).

Note that the threat model and assumptions are

- HBCs are NSB channels
- The user and all the nodes are trustworthy.
- The notification/verification signal (e.g. LED alarm signal that should be read by human users) is transmitted over a NSB HCC.
- Button pushing procedure is a NSB HCC.

Protocol 7.4 is the ordinary authorization method. The owner can use this protocol for permissions assignment. Permissions such as data access and device configuration correspond to different shared session keys between S and M.

In order to have a better resilience against DoS attacks, we designed a DoS prevention protocol as an assistant to the authorization protocol. The main idea is that C gives an alarm to the user whenever it detects a message that begins with msg_4. When the user hears the alarm, and if the user has not initiated a protocol run, he/she should immediately look for illegal HBC transmitters that are attached to his/her body. The assistant protocol works as below, and the message flow is shown in Protocol 7.5.

Step 0. 1 Whenever a message that begins with label msg_4 has been detected from HBCs, C goes to step 0.2.

Step 0.2 C gives an alarm to the user.

Step 0.3 If the protocol is initiated by the legal user, the user sends a yes signal to C via human-controlled channels; otherwise, the user sends a no signal to C via human-controlled channels.

Protocol 7.5 (DoS prevention protocol)**.**

0.1 $* \Longrightarrow \mathsf{C} : msg_4$

0.2 $\mathsf{C} \Longrightarrow$ User : alarm

0.3 User $\Longrightarrow \mathsf{C}$: yes/no

Note that the threat model and assumptions are

- HBCs are NSB channels

- The user and all the nodes are trustworthy.
- The notification/verification signal (e.g. LED alarm signal that should be read by human users) is transmitted over a HCC.
- Button pushing procedure is a HCC.

Accidental authorization We can also use PWD, C' and PWD' for authorization in some exceptional cases.

- PWD. If C is down, the owner can use PWD for authorization. This process is a password-based protocol, which is used in everyday life, so we will not elaborate the process here.
- C'. It is used in the following case: PWD is forgotten by the owner, and C is broken. In this case, the owner should prepare a new C; and then, the owner finds the emergency contact in order to establish a shared key between the new C and S. This key establishment process is similar to that of Protocol 7.4. After this process, the new PWD should be set.
- PWD'. This is used in emergency situations: the owner is unconscious, and C is down. In this case, the health authority can contact the emergency contact, and get PWD' in order to establish security links between the health authority's devices and S. This process is also a password-based protocol. Besides, PWD' should be updated using C' after the emergency.

7.3.5 Discussion

Advantages of HBSPs The first advantage is that HBSPs support implants, and are thus ideal for small-scale personal networks with some implanted nodes. In addition, HBC devices can be very small, and are thus suitable for extremely small nodes, i.e. implants.

The second advantage is that HBSPs can be quite straightforward. Users just need to touch the ports during the communications; they do not need to compare strings/digits/flash

patterns. In addition, this means that they can be easily understood by developers, and be securely implemented in real-world applications.

Disadvantages The main disadvantage is safety concerns. One potential danger is that the voltage or current becomes higher than the safe limit; thus, nodes and transceivers must be carefully designed before they go on the market.

Heat The other potential problem is that tissue absorbs the heat generated during HBC communications; this can be mitigated by using short digest instead of long public keys. However, wireless radio communications also generate heat. We have not found reports that compare the heat generated using the two communication technologies. Thus, whether we should use more wireless radio communications or HBCs is still an open issue.

7.4 Chapter Summary

In this chapter, HBCs, which are established based on IBCs, have been studied as one type of OOB channel. In addition, we have proposed HBSPs, which can bootstrap security in personal networks using HBCs.

IBCs are promising technologies in personal networks. We have investigated present day IBC devices and communication techniques. In addition, we have designed and built a demo IBC for evaluation. According to our experiment results, IBCs can be established using very simple devices and techniques; and they are fast.

HBCs are combinations of human-controlled channels and IBCs. Many HBCs are NSB channels, and conditional confidential channels. They are a great foundation of multi-channel security protocols.

HBSPs have been proposed. In HBSPs, sensor data and instructions can be directly exchanged via HBCs; this method is very straightforward. In addition, HBSPs for implants have been discussed, and example protocols have been designed.

In summary, the main advantage of HBSPs is that they can *provide security to implants*. In addition, HBSPs can reduce the burden of human users and support smaller nodes, which are an improvement over HISPs.

Chapter 8

Use Cases: Intra-body Security Protocols

This chapter contains two HBSP[1] use cases. These use cases will show that HBSPs can be building blocks of future security mechanisms in personal networks.

The first is the SecPN, which use HBSPs in a wearable personal network in order to bootstrap the network securely. This system shows that HBSPs are useful in personal networks: in reducing the human burden, and in reducing computation consumption in slave nodes.

The second one is revised Hölbl and Welzer identity-based key establishment protocols. We use HBSPs in order to eliminate man-in-the-middle and impersonation attacks; these attacks are found by Shim [100] against an identity-based protocol proposed in [48].

Contents

[1] HB for HBC, and SP for security protocol.

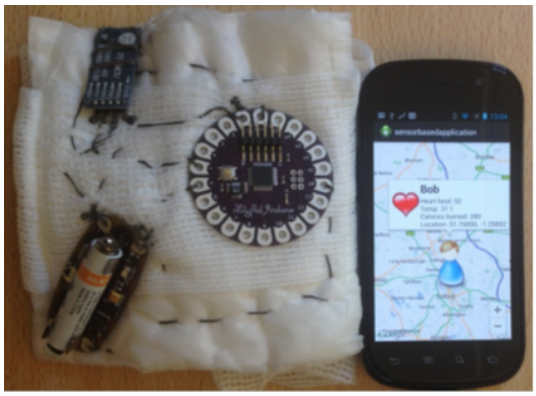


Figure 8.1: SecPN

8.1 SecPN

This section includes (1) the demo system SecPN and related security protocols, and (2) discussion.

8.1.1 Demo System and Security Protocols

The SecPN demo system is shown in Fig. 8.1. The coordinator is a mobile phone, and the slave node is a Lilipad board, which is sewn onto a bandage using conductive thread. A Bluetooth module, a power source and a temperature sensor (on the back side of the bandage) are also sewn onto the bandage, and are connected to the Lilipad board.

Note that the threat model and assumptions are

- HBCs are NSB channels (in SecPN V, HBCs are NSB and CON channels)
- The user and all the nodes are trustworthy.
- group size is inputted via a NSB HCC.
- The notification/verification signal (e.g. LED alarm signal that should be read by human users) is transmitted over a NSB HCC.
- Button pushing procedure is a NSB HCC.

SecPN I The first security protocol used in SecPN is named the SecPN I. It is a key transport protocol, and it can establish keys between C and each S_i separately. The message

flow is shown in Protocol 8.1. The details are explained below.

Protocol 8.1 (SecPN I)**.**

1. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : \mathsf{C}, PK_{\mathsf{C}}$
2. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} : \mathsf{S}_i, encr_{PK_{\mathsf{C}}}(SHK_i)$
3. $\forall \mathsf{S}_i \Longrightarrow \mathsf{C} : \mathsf{S}_i, H_i$

Step 1. (a) The user inputs group size gs into C, and pushes a button in each node in order to initiate the protocol. (b) After this, C broadcasts its identity C and its public key PK_{C} over wireless radio channels.

Step 2. (a) After receiving message 1, each slave node S_i replies to message 2 via wireless radio channels. This message includes its identity S_i, and $encr_{PK_{\mathsf{C}}}(SHK_i)$, which is the shared key SHK_i between S_i and C encrypted using PK_{C}. (b) If C has received $gs - 1$ message 2, it goes to step 3; otherwise, it aborts and informs the user that the protocol has failed.

Step 3. (a) Each S_i sends its identity S_i and $H_i = hash(\mathsf{C}, PK_{\mathsf{C}}, \mathsf{S}_i, SHK_i)$ to C over HBCs. Each S_i delays $150i$ time unit. (b) C verifies the received H_i. C sends the verification result to each S_i and the user over human-controlled channels.

SecPN II As the second option, the user can run SecPN II for transporting keys. The message flow is shown in Protocol 8.2. The details are explained below.

Protocol 8.2 (SecPN II)**.**

1. $\forall \mathsf{S}_i \Longrightarrow \mathsf{C} : \mathsf{S}_i, PK_i$
2. $\mathsf{C} \Longrightarrow \forall \mathsf{S}_i : \mathsf{C}, \mathsf{S}_i, encr_{PK_i}(SHK_i)$

Step 1. (a) The user inputs group size gs into C, and pushes a button on each node in order to initiate the protocol. (b) S_1 broadcasts its identity S_i and public key PK_i over HBCs.

Step 2. (a) If C has received $gs - 1$ message 1, it goes to step 2.b; otherwise, it aborts and informs the user that the protocol has failed. (b) C sends S_i message 2 over HBCs. This message includes its identity C, objective node identity S_i, and $encr_{PK_i}(SHK_i)$, which

is the shared key SHK_i between S_i and C encrypted using PK_i.

SecPN III We can exchange ECDH public tokens using SecPN III. The message flow is shown in Protocol. 8.3. The details are explained below.

Protocol 8.3 (SecPN III)**.**

1. $\mathsf{C} \xrightarrow{\mathsf{C},gs,EU_{\mathsf{C}}} \mathsf{S}_i$
2. $\mathsf{C} \xleftarrow{\mathsf{S}_i,EU_i} \mathsf{S}_i \xrightarrow{\mathsf{S}_i,EU_i} \mathsf{S}_j$
3. $\mathsf{C} \xRightarrow{H} \mathsf{S}_i$

Step 1. (a) The user inputs group size gs on C, and pushes a button on each node in order to initiate the protocol. (b) After this, C broadcasts its identity C, group size gs, and public token EU_{C} over wireless radio channels.

Step 2. (a) S_i broadcasts its identity S_i, and public token EU_i over wireless radio channels. (b) Each node checks that the group size is gs, otherwise its buzzer gives an alarm.

Step 3. (a) C computes $H = hash(\mathsf{C}, EU_{\mathsf{C}}, \sum(\mathsf{S}_i, EU_i), gs)$. C broadcasts its H over HBCs. (b) S_i verifies the received H. If the verification fails, the protocol fails, and an alarm is generated using a buzzer. Otherwise, all the nodes have exchanged the ECDH public tokens successfully.

SecPN IV SecPN IV is a straightforward parallel key agreement protocol. The message flow is shown in Protocol 8.4; and the details are explained below.

Protocol 8.4 (SecPN IV)**.**

1. $\mathsf{C} \Longrightarrow \forall\, \mathsf{S}_i : \mathsf{C}, EU_{\mathsf{C}}$
2. $\forall\, \mathsf{S}_i \Longrightarrow \mathsf{C} : \mathsf{S}_i, EU_i$

Step 1. (a) The user inputs group size gs into C, and pushes a button on each node in order to initiate the protocol. (b) After button pushing, C broadcasts its identity C and its public token EU_{C} over HBCs.

Table 8.1: SecPN Protocols Comparison

	AsyC	AsyS
SecPN I	$gs-1$	$gs-1$
SecPN II	$gs-1$	$gs-1$
SecPN III	$gs-1$	$(gs-1)(gs-1)$
SecPN IV	$gs-1$	$gs-1$
SecPN V	0	0

Step 2. (a) After receiving message 1, S_i replies with its identity S_i, and public token EU_i over HBCs. (b) C counts the number of message 2. If the number is $gs-1$, the protocol succeeds; otherwise, the protocol fails.

SecPN V As the last option, the user can run SecPN V for transporting symmetric keys. The message flow is shown in Protocol 8.5. The details are explained as follows.

Protocol 8.5 (SecPN V)**.**

1. $\mathsf{C} \Longrightarrow \forall\, \mathsf{S}_i : \mathsf{C}, \mathsf{S}_i, SHK_i$

Step 1. The user inputs group size gs into C, and pushes buttons on each node in order to initiate the protocol.

Step 2. After button pushing, C generates $gs-1$ different symmetric keys SHK_i. C sends its identity C and SHK_i to S_i over HBCs.

This protocol is especially suitable for implants, because it does not involve expensive asymmetric cryptographic algorithms. However, this protocol relies on conditional confidentiality of NSB HBC channels, which is a strong assumption; thus users must be alert to the environment, e.g. stay away from other people at least a distance of 1 metre.

8.1.2 Discussion

These protocols used in SecPN are straightforward. In addition, in all these protocols, human users need only to input group sizes, to push buttons, and to prevent physical attacks; digest or hash comparisons are no longer required.

Besides, their computation consumption is roughly compared in Table 8.1. AsyC is the number of asymmetric key cryptography computations in the coordinator; and AsyS is the

total number of asymmetric key cryptography computations in slave nodes. As we can see, SecPNV is the most computation-efficient protocol, but its security assumption is the strongest.

8.2 Multi-Channel Identity-based Protocols

This section includes (1) a review of Hölbl and Welzer protocol; (2) a review of attacks; (3) HBC-based HW protocols; and (4) discussions.

8.2.1 Review of Hölbl and Welzer Protocol

Identity-based infrastructure allows the public key of one user to be easily derived from her known identity, for example, her phone number and node universally unique identifier (UUID). Such an infrastructure alleviates the certificate overhead using a public key infrastructure.

Hölbl and Welzer [48] have proposed an identity-based key establishment protocol, which is named HW in this thesis. HW consists of three phases: system setup, key extraction, and key agreement. They are introduced below.

System setup. In this phase, several private and public values are generated by a trusted key generation centre KGC.

- Private value: a secret random integer X;
- Public values: a large prime integer p, a primitive root g, a one-way function $hash()$, and $Y = g^X (mod\, p)$

Key extraction. A pair of public and private keys is generated in KGC for each node. These keys are pre-distributed to each node.

- For coordinator node C:
  - Hash of its identity $I_{\mathsf{C}} = hash(\mathsf{C})$
  - A random number K_{C}
  - A public key $U_{\mathsf{C}} = g^{K_{\mathsf{C}}} (mod\, p)$

– A private key $V_{\mathsf{C}} = I_{\mathsf{C}} K_{\mathsf{C}} + X U_{\mathsf{C}} (mod\, p-1)$

- For each slave node S_i:
  - Hash of its identity $I_i = hash(\mathsf{S}_i)$
  - A random number K_i
  - A public key $U_i = g^{K_i} (mod\, p)$
  - A private key $V_i = I_i K_i + X U_i (mod\, p-1)$

Key agreement. The message flow is shown in Protocol 8.6. The details are explained below.

Protocol 8.6 (HW)**.**

1. $\mathsf{C} \longrightarrow \mathsf{S}_i : \mathsf{C}, U_{\mathsf{C}}, T_{\mathsf{C}}$
2. $\mathsf{S}_i \longrightarrow \mathsf{C} : \mathsf{S}_i, U_i, T_i$

Step 1. It consists of the following substeps.

1.1 C selects a random number R_{C};

1.2 C computes $T_{\mathsf{C}} = g^{R_{\mathsf{C}}}$;

1.3 C sends $\{\mathsf{C}, U_{\mathsf{C}}, T_{\mathsf{C}}\}$ to S_i;

1.4 S_i computes $I_{\mathsf{C}} = hash(\mathsf{C})$;

1.5 S_i computes $g^{V_{\mathsf{C}}} = U_{\mathsf{C}}^{I_{\mathsf{C}}} \cdot Y^{U_{\mathsf{C}}}$;

1.6 S_i computes $SHK_i = (g^{V_{\mathsf{C}}} \cdot T_{\mathsf{C}})^{V_i + R_i} = g^{(R_{\mathsf{C}} + V_{\mathsf{C}})(R_i + V_i)}$.

Step 2. It consists of the following substeps.

2.1 S_i selects a random number R_i;

2.2 S_i computes $T_i = g^{R_i}$;

2.3 S_i sends $\{\mathsf{S}_i, U_i, T_i\}$ to C;

2.4 C computes $I_i = hash(\mathsf{S}_i)$;

2.5 C computes $g^{V_i} = U_i^{I_i} \cdot Y^{U_i}$;

2.6 C computes $SHK_i = (g^{V_i} \cdot T_i)^{V_\mathsf{C}+R_\mathsf{C}} = g^{(R_i+V_i)(R_\mathsf{C}+V_\mathsf{C})}$.

8.2.2 Review of Attacks

However, Shim [100] recently found that HW is vulnerable to MITM attack and to impersonation attack.

MITM attack HW is vulnerable to MITM attack. Attacker A can generate two random values α and β; and, as the result of the attack, C and A have a shared key $g^{(\beta)(R_\mathsf{C}+V_\mathsf{C})}$; in addition, S_i and A have a shared key $g^{(\alpha)(R_i+V_i)}$. The message flow of the MITM attack is shown as Attack 8.1. The steps are explained below.

Step 1. Attacker A intercepts messages from C to S_i. A receives $\{\mathsf{C}, U_\mathsf{C}, T_\mathsf{C}\}$. A computes the shared key between $\mathsf{A}(\mathsf{S}_i)$ and C: $(g^{V_\mathsf{C}} \cdot T_\mathsf{C})^{\beta} = g^{\beta(R_\mathsf{C}+V_\mathsf{C})}$.

Step 2. A sends $\{\mathsf{C}, U_\mathsf{C}, g^{\alpha-V_\mathsf{C}}\}$ to S_i. S_i computes the shared key between $\mathsf{A}(\mathsf{C})$ and S_i: $(g^{V_\mathsf{C}} \cdot g^{\alpha-V_\mathsf{C}})^{V_i+R_i} = g^{\alpha(R_i+V_i)}$.

Step 3. Attacker A intercepts messages from S_i to C. A receives $\{\mathsf{S}_i, U_i, T_i\}$. A computes the shared key between $\mathsf{A}(\mathsf{C})$ and S_i: $(g^{V_i} \cdot T_i)^{\alpha} = g^{\alpha(R_i+V_i)}$.

Step 4. A sends $\{\mathsf{S}_i, U_i, g^{\beta-V_i}\}$ to C. C computes the shared key between $\mathsf{A}(\mathsf{S}_i)$ and C: $(g^{V_i} \cdot g^{\beta-V_i})^{V_\mathsf{C}+R_\mathsf{C}} = g^{\beta(R_\mathsf{C}+V_\mathsf{C})}$.

Attack 8.1 (HW-MITM)**.**

1. $\mathsf{C} \longrightarrow \mathsf{A} : \mathsf{C}, U_\mathsf{C}, T_\mathsf{C}$
2. $\mathsf{A} \longrightarrow \mathsf{S}_i : \mathsf{C}, U_\mathsf{C}, g^{\alpha-V_\mathsf{C}}$
3. $\mathsf{S}_i \longrightarrow \mathsf{A} : \mathsf{S}_i, U_i, T_i$
4. $\mathsf{A} \longrightarrow \mathsf{C} : \mathsf{S}_i, U_i, g^{\beta-V_i}$

Impersonation attack HW is vulnerable to impersonation attack. Attacker A can successfully impersonate C to S_i, and they have a shared key $g^{t(R_i+V_i)}$. The attack is shown in Attack 8.2, and the steps are explained below.

Attack 8.2 (HW-impersonation)**.**

1. $\mathsf{A} \longrightarrow \mathsf{S}_i : \mathsf{C}, U_{\mathsf{C}}, g^{R_{\mathsf{C}}}$
2. $\mathsf{S}_i \longrightarrow \mathsf{A} : \mathsf{S}_i, U_i, T_i$

Step 1. (a) A computes $g^{R_{\mathsf{C}}} = g^t / g^{V_{\mathsf{C}}} = g^t / (U_{\mathsf{C}}^{I_{\mathsf{C}}} \cdot Y^{U_{\mathsf{C}}})$, where t is a random value chosen by A. (b) A sends $\{\mathsf{C}, U_{\mathsf{C}}, g^{R_{\mathsf{C}}}\}$ to S_i. S_i gets the shared key: $(g^{V_{\mathsf{C}}} \cdot g^{R_{\mathsf{C}}})^{V_i + R_i} = g^{t(R_i+V_i)}$.

Step 2. S_i sends $\{\mathsf{S}_i, U_i, T_i\}$ to A. A computes the shared key: $(g^{V_i} \cdot T_i)^t = g^{t(R_i+V_i)}$.

8.2.3 HBC-based HW Protocols

HBSP-HW I HBSP-HW I is a straightforward revised protocol of HW. It exchanges all the HW messages via HBCs.

Protocol 8.7 (HBSP-HW I)**.**

1. $\mathsf{C} \Longrightarrow_{HBC} \forall \mathsf{S}_i : \mathsf{C}, U_{\mathsf{C}}, T_{\mathsf{C}}$
2. $\forall \mathsf{S}_i \Longrightarrow_{HBC} \mathsf{C} : \mathsf{S}_i, U_i, T_i$

Step 1. (a) The user pushes a button on each node in order to initiate the protocol. (b) The user inputs group size gs into C via human-controlled channels. (c) After gs has been received, C broadcasts $\{\mathsf{C}, U_{\mathsf{C}}, T_{\mathsf{C}}\}$ over HBCs.

Step 2. (a) After receiving message 1, slave node S_i replies to message 2 $\{\mathsf{S}_i, U_i, T_i\}$ over HBCs. (b) After receiving message 2, coordinator node C informs the user of the result of this protocol. If the number of received message 2 in C is $gs - 1$, C tells the user that the protocol has succeeded; otherwise it tells the user that the protocol has failed.

HBSP-HW II In order to reduce the safety concerns of using HBCs, we can reduce the messages transmitted via HBCs by using hash functions.

Step 1. (a) The user pushes a button on each node in order to initiate the protocol. (b) The user inputs group size gs on C via human-controlled channels. (c) After gs has been received, C broadcasts $\{\mathsf{C}, U_{\mathsf{C}}, T_{\mathsf{C}}\}$ over wireless radio channels.

Step 2. (a) After message 1 has been received, each S_i broadcasts $\{\mathsf{S}_i, U_i, T_i\}$ over wireless radio channels. (b) Each slave node records the number of received messages. (c)

Protocol 8.8 (HBSP-HW II)**.**

1. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : \mathsf{C}, U_{\mathsf{C}}, T_{\mathsf{C}}$
2. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} \wedge \forall \mathsf{S}_j : \mathsf{S}_i, U_i, T_i$
3. $\mathsf{C} \Longrightarrow_{HBC} \forall \mathsf{S}_i : H$
4. $\forall \mathsf{S}_i \Longrightarrow_{HBC} \mathsf{C} : \mathsf{S}_i, \text{yes/no}$

C counts the number of received messages. If the number is $gs - 1$, it goes to step 3.

Step 3. C broadcasts $H = hash(gs, \mathsf{C}, U_{\mathsf{C}}, T_{\mathsf{C}}, \sum(\mathsf{S}_i, U_i, T_i))$ over HBCs.

Step 4. (a) Each S_i verifies the received H. (b) Each S_i sends C the verification result yes/no via HBCs. (c) If C has received yes signals from all the $gs - 1$ slave nodes, C sends a notification of success to the user via human-controlled channels.

HBSP-HW III We can further reduce the amount of bits transmitted over HBCs by using Protocol 8.9. In this protocol, we need only transmit 16-bit digest outputs through the human body instead of 160-bit hash outputs.

Protocol 8.9 (HBSP-HW III)**.**

1. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : \mathsf{C}, U_{\mathsf{C}}, T_{\mathsf{C}}, HC_{\mathsf{C}}, gs$
2. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} \wedge \forall \mathsf{S}_j : \mathsf{S}_i, U_i, T_i, HC_i$
3. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : N_{\mathsf{C}}$
3. $(\forall \mathsf{S}_i \Longrightarrow_{HBC} \mathsf{C} : \mathsf{S}_i, \text{no})$
4. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} \wedge \forall \mathsf{S}_j : N_i$
5. $\mathsf{C} \Longrightarrow_{HBC} \forall \mathsf{S}_i : D$
6. $\forall \mathsf{S}_i \Longrightarrow_{HBC} \mathsf{C} : \mathsf{S}_i, \text{yes/no}$

Step 1. (a) *bp*: the user pushes a button on each node in order to initiate the protocol. (b) The user inputs group size gs on C via human-controlled channels. (b) After gs has been received, C generates a long random number N_{C} and $HC_{\mathsf{C}} = hash(N_{\mathsf{C}}, \mathsf{C})$. C broadcasts $\{\mathsf{C}, U_{\mathsf{C}}, T_{\mathsf{C}}, HC_{\mathsf{C}}, gs\}$ over wireless radio channels.

Step 2. (a) After message 1 has been received, each S_i generates a long random number N_i and $HC_i = hash(N_i, \mathsf{S}_i)$. Each S_i broadcasts $\{\mathsf{S}_i, U_i, T_i, HC_i\}$ over wireless radio chan-

nels. (b) *cp*: each node counts the number of received different commitment messages. If the number is $gs-1$, it ceases to accept any new commitment message, and it goes to step 3.

Step 3. (a) *rp*: C sends the opening message N_{C} to each S_i over wireless radio channels. (b) Each S_i verifies that $HC_{\mathsf{C}} = hash(N_{\mathsf{C}}, \mathsf{C})$. If the verification succeeds, it goes to step 4; otherwise, it sends a failure message to C via HBCs.

Step 4. (a) Each S_i broadcasts its opening message N_i over wireless radio channels. (b) C verifies that $HC_i = hash(N_i, \mathsf{S}_i)$. If the verification fails, C aborts and informs the user via human-controlled channels. If the verification succeeds, it goes to step 5.

Step 5. C broadcasts $D = digest(N_{\mathsf{C}} \oplus (\bigoplus N_i), gs, \mathsf{C}, U_{\mathsf{C}}, T_{\mathsf{C}}, \sum(\mathsf{S}_i, U_i, T_i))$ via HBCs.

Step 6. (a) Each S_i sends C the verification result of the received D. (b) *ep*: if C has received yes signals from all the $gs-1$ slave nodes, C sends a notification of success to the user via human-controlled channels; otherwise, C informs the user that the protocol has failed.

Remark The threat model and assumptions are

- HBCs are NSB channels
- The user and all the nodes are trustworthy.
- group size is inputted via a NSB HCC.
- The notification/verification signal (e.g. LED alarm signal that should be read by human users) is transmitted over a NSB HCC.
- Button pushing procedure is a NSB HCC.

These protocols are mainly designed for wearable networks. The nodes must have ports that can transmit and receive electrical signals.

8.2.4 Discussion

Using HBC-based HW protocols, impersonation and MITM attacks can be eliminated. This is achieved because HBC-based HW protocols can guarantee the authenticity and integrity

of HW messages.

Impersonation attack is eliminated. The main reason is that the following attacks can be eliminated: the attacker A transmits illegal key messages via HBCs in HBSP-HW I; A transmits illegal hash outputs in HBSP-HW II; or A transmits illegal digest outputs in HBSP-HW III.

MITM attack is also eliminated. Firstly, in HBSP-HW I, the key exchange messages are transmitted over HBC channels; thus attacks that change these messages can be eliminated. Secondly, in HBSP-HW II, if A changes the HW messages, this can be found by checking the hash outputs transmitted over HBCs. Finally, in HBSP-HW III, the illegally changed HW messages can be found by checking digest outputs transmitted over HBCs.

8.3 Chapter Summary

In this chapter, we have studied two HBSP use cases. These use cases show that HBSPs are useful in personal networks.

Firstly, HBSPs are useful in the key establishment process. SecPN is a wearable personal network that includes one mobile phone and several wearable slave nodes. We designed several HBSPs that can help these nodes to establish keys. These protocols have the following advantages. First of all, they can reduce the burden of human users by using HBCs. In addition, they can reduce the computation consumption in slave nodes.

Secondly, HBSPs are useful in identity-based security systems. An identity-based system allows the public key of one node to be easily derived from its known identity, for example node UUID. Such an infrastructure alleviates the certificate overhead by using public key infrastructures. Hölbl and Welzer [48] have proposed an identity-based key establishment protocol, named HW. However, Shim [100] recently found that HW is vulnerable to MITM attack and impersonation attack. In order to eliminate these attacks, we have designed several HBC-based protocols. This use case shows that HBCs are a good building block of security protocols.

In summary, these use cases clearly show that HBSPs are promising. They can be building blocks of future security mechanisms, especially mechanisms in future wearable

and implanted computing systems.

Chapter 9

Protocol Evaluation

In this chapter, we will evaluate multi-channel security protocols mentioned in previous chapters. We will evaluate the following metrics: energy consumption due to wireless radio communications; cryptographic computations; comparisons of key schemes; and security discussions.

Contents

9.1 Wireless Radio Communications

9.1.1 Simulation Preparation

In personal networks, it is reported that wireless radio communications are the dominant energy consumption processes [128]. In order not to get distracted by the details of different OOB channels, we use tailored multi-channels protocols[1]; the correspondence between the tailored protocols and their message flows is shown in Table 9.1.

[1] G for group; T for direct transmission of $Info$; P for parallel; H for hash; HCBK and SHCBK are protocol models mentioned many times in previous chapters

Table 9.1: Tailored protocols and implementations

Protocol	Message flow	Implementation
GT	Protocol 9.1	
PT	Protocol 9.2	
GH	Protocol 9.3	Fig. 9.1
PH	Protocol 9.4	Fig. 9.2
GHCBK	Protocol 9.5	Fig. 9.3
PHCBK	Protocol 9.6	Fig. 9.4
GSHCBK	Protocol 9.7	Fig. 9.5
PSHCBK	Protocol 9.8	Fig. 9.6

To examine the energy consumption of tailored protocols, simulations have been done in the Cooja simulator [86] using Tmote Sky nodes. The actions of nodes are mainly controlled in a thread. In this thread, we trace the power consumption in every 5 time units. After this, the broadcast channel is open. The communication steps of our protocols are implemented in a while loop. The correspondence between the tailored protocols and their implementations is shown in Table 9.1.

Protocol 9.1 (GT)**.**

1. $\mathsf{C} \Longrightarrow \forall \mathsf{S}_i : \mathsf{C}, Info_{\mathsf{C}}$
2. $\forall \mathsf{S}_i \Longrightarrow \mathsf{C} \wedge \forall \mathsf{S}_j : \mathsf{S}_i, Info_i$

Protocol 9.2 (PT)**.**

1. $\mathsf{C} \Longrightarrow \forall \mathsf{S}_i : \mathsf{C}, Info_{\mathsf{C}}$
2. $\forall \mathsf{S}_i \Longrightarrow \mathsf{C} : \mathsf{S}_i, Info_i$

Protocol 9.3 (GH)**.**

1. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : \mathsf{C}, Info_{\mathsf{C}}$
2. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} \wedge \forall \mathsf{S}_j : \mathsf{S}_i, Info_i$
3. $\mathsf{C} \Longrightarrow \forall \mathsf{S}_i : H$

Protocol 9.4 (PH)**.**

1. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : \mathsf{C}, Info_{\mathsf{C}}$
2. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} : \mathsf{S}_i, Info_i$
3. $\mathsf{C} \Longrightarrow \forall \mathsf{S}_i : H_i$

Protocol 9.5 (GHCBK)**.**

1. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : \mathsf{C}, Info_{\mathsf{C}}, HC_{\mathsf{C}}$
2. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} \wedge \forall \mathsf{S}_j : \mathsf{S}_i, Info_i$
3. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : N_{\mathsf{C}}$
4. $\mathsf{C} \Longrightarrow \forall \mathsf{S}_i : D$

Protocol 9.6 (PHCBK)**.**

1. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : \mathsf{C}, Info_{\mathsf{C}}, HC_{\mathsf{C}}$
2. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} : \mathsf{S}_i, Info_i$
3. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : N_{\mathsf{C}}$
5. $\mathsf{C} \Longrightarrow \forall \mathsf{S}_i : D_i$

Protocol 9.7 (GSHCBK)**.**

1. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : \mathsf{C}, Info_{\mathsf{C}}, HC_{\mathsf{C}}$
2. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} \wedge \forall \mathsf{S}_j : \mathsf{S}_i, Info_i, HC_i$
3. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : N_{\mathsf{C}}$
4. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} \wedge \forall \mathsf{S}_j : N_i$
5. $\mathsf{C} \Longrightarrow \forall \mathsf{S}_i : D$

9.1.2 Experiments

Here, we mainly test the energy consumptions of different protocols with different group sizes. Note that GT and PT do not use wireless radio communications; thus they are not involved. The parameters of the experiments and results (in Fig. 9.7) are explained in Table 9.2. The results are further elaborated below.

From Fig. 9.7, we can get the following results regarding wireless radio communications.

Protocol 9.8 (PSHCBK)**.**

1. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : \mathsf{C}, Info_{\mathsf{C}}, HC_{\mathsf{C}}$
2. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} : \mathsf{S}_i, Info_i, HC_i$
3. $\mathsf{C} \longrightarrow \forall \mathsf{S}_i : N_{\mathsf{C}}$
4. $\forall \mathsf{S}_i \longrightarrow \mathsf{C} : N_i$
5. $\mathsf{C} \Longrightarrow \forall \mathsf{S}_i : D_i$

```
PROCESS_THREAD(example_broadcast_process, ev, data)
{
  /* while loop, which includes the communication steps*/
  while(1){
    if (this node is a coordinator node, and message 1 has not been sent){
    generate and broadcast message 1;
    }
    if (this node is a slave node, message 2 has not been sent, and message 1
        has been received){
    generate and broadcast message 2;
    }
    if (this node is a coordinator node, message 1 has been sent, and the
        number of received message 2 equals to gs-1){
    generate and show digest;
    }
    if (this node is a slave node, message 2 has been sent, message 1 has been
         received, and the number of received message 2 equals to gs-2){
    generate and show digest;
    }
  }
}
```

Figure 9.1: Example of the control thread of GH protocol

Table 9.2: Parameters of the experiments and results

Node transmission range	50 meters
Group size gs	5 and 10
Network topology	full mesh
Horizontal axis in Fig 9.7	group size gs
Vertical axis in Fig 9.7	normalized energy consumption
5 in Fig 9.7	average energy consumption in each node, $gs = 5$
10 in Fig 9.7	average energy consumption in each node, $gs = 10$
5s in Fig 9.7	average energy consumption in each slave node, $gs = 5$
10s in Fig 9.7	average energy consumption in each slave node, $gs = 10$

With the same group size, these protocols can be sorted in ascending order by the energy consumptions as follows: (1) hash-based protocols: GH and PH, (2) HCBK protocols: GHCBK and PHCBK, (3) SHCBK protocols: GSHCBK and PSHCBK. The corresponding

```
PROCESS_THREAD(example_broadcast_process, ev, data)
{
  while(1){
    if (this node is a coordinator node, and message 1 has not been sent){
    generate and broadcast message 1;
    }
    if (this node is a slave node, message 2 has not been sent, and message 1
        has been received){
    generate and broadcast message 2;
    }
    if (this node is a coordinator node, message 1 has been sent, and the
        number of received message 2 equals to gs-1){
    generate and show digest;
    }
    if (this node is a slave node, message 2 has been sent, message 1 has been
         received){
    generate and show digest;
    }
  }
}
```

Figure 9.2: Example of the control thread of PH protocol

```
PROCESS_THREAD(example_broadcast_process, ev, data)
{
  while(1){
    if (this node is a coordinator node, and message 1 has not been sent){
    generate and broadcast message 1;
    }
    if (this node is a slave node, message 2 has not been sent, and message 1
        has been received){
    generate and broadcast message 2;
    }
    if (this node is a coordinator node, message 3 has not been sent, and the
        number of received message 2 equals to gs-1){
    generate and broadcast message 3;
    }
    if (this node is a coordinator node, message 1 and message 3 have been
        sent, and the number of received message 2 equals to gs-1){
    generate and show digest;
    }
    if (this node is a slave node, message 2 has been sent, message 1 and
        message 3 have been received, and the number of received message 2
        equals to gs-2){
    generate and show digest;
    }
  }
}
```

Figure 9.3: Example of the control thread of GHCBK protocol

ranking of the protocols in previous chapters is listed in Table 9.3.

Additionally, in SHCBK protocols and hash-based protocols, the parallel versions are

```
PROCESS_THREAD(example_broadcast_process, ev, data)
{
  while(1){
    if (this node is a coordinator node, and message 1 has not been sent){
    generate and broadcast message 1;
    }
    if (this node is a slave node, message 2 has not been sent, and message 1
        has been received){
    generate and broadcast message 2;
    }
    if (this node is a coordinator node, message 3 has not been sent, message
        1 has been sent, and the number of received message 2 equals to gs-1){
    generate and broadcast message 3;
    }
    if (this node is a coordinator node, message 1 and message 3 have been
        sent, and the number of received message 2 equals to gs-1){
    generate and show digest;
    }
    if (this node is a slave node, message 2 has been sent, message 1 and
        message 3 have been received){
    generate and show digest;
    }
  }
}
```

Figure 9.4: Example of the control thread of PHCBK protocol

Table 9.3: Ranking I

Hash-based protocols	HVSP-L I in Chapter 6, SecPN I, III, HBSP-HW II in Chapter 8
HCBK protocols	HISP-CI in Chapter 3, HISP-K I, HISP-B I and II in Chapter 4, HVSP-CI in Chapter 5, HVSP-K I, III in Chapter 6
SHCBK protocols	HISP-SD in Chapter 3, HISP-K II in Chapter 4, HVSP-SD in Chapter 5, HVSP-K II, IV in Chapter 6, HBSP-HW III in Chapter 8

better than the group versions; more clearly, PSHCBK is better than GSHCBK, and PH is better than GH. The corresponding ranking of the protocols in previous chapters is listed in Table 9.4. Note that there are no significant differences between PHCBK and GHCBK.

Besides, we find that the energy consumption cannot be simply modelled as message numbers or the number of sent and received messages. For example, in Fig. 9.8(a), GHtest is the experiment result of GH; GHmsg is the message number of GH; and GHsr is the number of sent and received messages in GH. In Fig. 9.8(b), PHtest is the experiment

```
PROCESS_THREAD(example_broadcast_process, ev, data)
{
  while(1){
    if (this node is a coordinator node, and message 1 has not been sent){
    generate and broadcast message 1;
    }
    if (this node is a slave node, message 2 has not been sent, and message 1
        has been received){
    generate and broadcast message 2;
    }
    if (this node is a coordinator node, message 3 has not been sent, and the
        number of received message 2 equals to gs-1){
    generate and broadcast message 3;
    }
    if (this node is a slave node, message 4 has not been sent, the number of
        received message 2 equals to gs-2, and message 3 has been received){
    generate and broadcast message 4;
    }
    if (this node is a coordinator node, message 1 and message 3 have been
        sent, and the numbers of received message 2 and message 4 equal to gs
        -1){
    generate and show digest;
    }
    if (this node is a slave node, message 2 and message 4 have been sent,
        message 1  and message 3 has been received, and the numbers of
        received message 2  and message 4 equal to gs-2){
    generate and show digest;
    }
  }
}
```

Figure 9.5: Example of the control thread of GSHCBK protocol

Table 9.4: Ranking II

PH	HVSP-L I in Chapter 6, SecPN I in Chapter 8
GH	SecPN III, HBSP-HW II in Chapter 8
PSHCBK	HVSP-SD in Chapter 5, HVSP-K II, IV in Chapter 6,
GSHCBK	HISP-SD in Chapter 3, HISP-K II in Chapter 4, HBSP-HW III in Chapter 8

result of PH; PHmsg is the message number of PH; and PHsr is the number of sent and received messages in PH. We can clearly see that neither the values nor the trends are the same, so message numbers or the number of sent and received messages are not good models of energy consumption caused by wireless radio communications.

Note that the discussion in this section is energy consumption related to wireless radio communications. The energy consumption related to OOB channels is not included. This is

```
PROCESS_THREAD(example_broadcast_process, ev, data)
{
  while(1){
    if (this node is a coordinator node, and message 1 has not been sent){
    generate and broadcast message 1;
    }
    if (this node is a slave node, message 2 has not been sent, and message 1
        has been received){
    generate and broadcast message 2;
    }
    if (this node is a coordinator node, message 3 has not been sent, and the
        number of received message 2 equals to gs-1){
    generate and broadcast message 3;
    }
    if (this node is a slave node, message 4 has not been sent, and message 3
        has been received){
    generate and broadcast message 4;
    }
    if (this node is a coordinator node, message 1 and message 3 have been
        sent, the numbers of received message 2 and message 4 equal to gs-1){
    generate and show digest;
    }
    if (this node is a slave node, message 2 and message 4 have been sent, and
         message 1 and message 3 have been received){
    generate and show digest;
    }
  }
}
```

Figure 9.6: Example of the control thread of PSHCBK

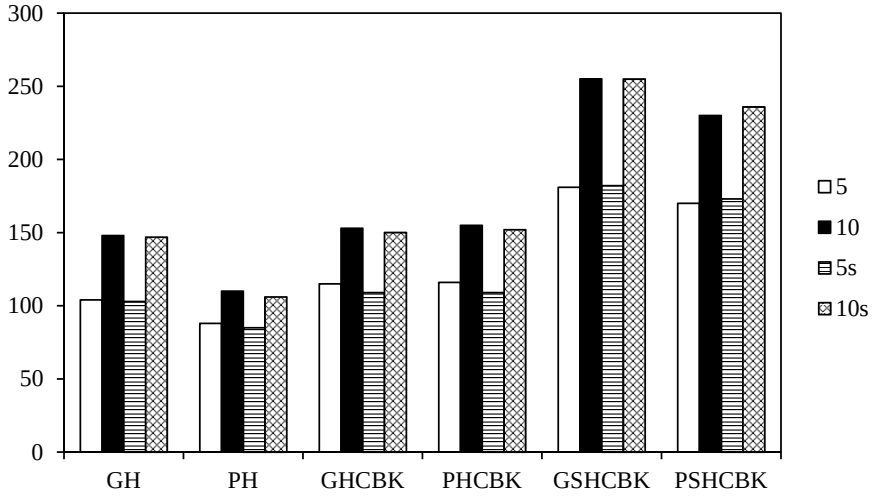
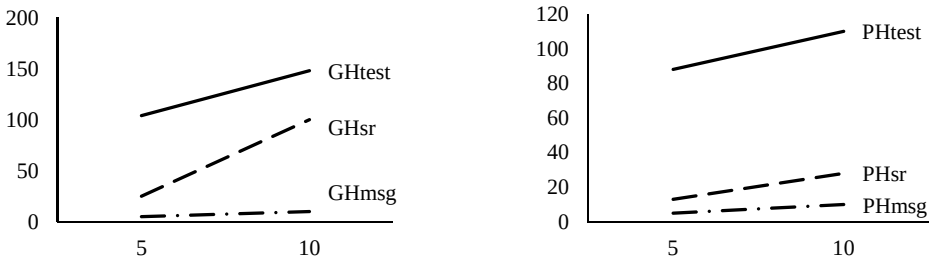


Figure 9.7: Communication efficiency comparison. The y axis is the normalized energy consumption.

because wireless radio communications are the most energy-consuming process [128, 22] in general. However, if some OOB channels are also energy-consuming, the previous rankings



(a) GH (b) PH

Figure 9.8: Communication model comparison. The y axis is the normalized energy consumption. The x axis is the group sizes.

Table 9.5: Key Schemes Comparison

	Storage	Connectivity	Flexibility	Resilience
HISP-K I	★	★★★	★★★	★★★
HISP-K II	★	★★★	★★★	★★★
HVSP-K I	★★★	★	★★★	★★
HVSP-K II	★★★	★	★★★	★★
HVSP-K III	★★	★★	★★★	★★
HVSP-K IV	★★	★★	★★★	★★
SecPN I	★★★	★	★★★	★★
SecPN II	★★	★★	★★★	★★
SecPN III	★	★★★	★★★	★★★
SecPN IV	★★★	★	★★★	★★
SecPN V	★★	★★	★★	★★
HBSP-HW I	★★★	★	★★★	★★
HBSP-HW II	★	★★★	★★★	★★★
HBSP-HW III	★	★★★	★★★	★★★

should be reconsidered.

9.2 Key Schemes

In Table 9.5, the key schemes are evaluated. The evaluation is based on four metrics: storage, connectivity, flexibility and resilience of key compromise. As we can see, all the protocols have their advantages and disadvantages.

Storage. The protocols with three stars store only one key in each slave node; this is good for the memory-limited nodes. The protocols with two stars are the key transport protocols, the number of keys in each slave nodes depending on the deployment adjacent matrix mentioned in Chapter 6; generally, the number is between 1 and $gs-1$. The protocols

with one star use pair-wise keys, they store $gs - 1$ keys in each slave node; however, since the network scale is small, they are still acceptable.

Connectivity. The protocols with three stars will generate full mesh networks. The protocols with two stars will generate partial mesh networks, generally speaking. The protocols with one star will generate star networks; however, in many real-world personal networks, star networks are good enough.

Flexibility. All of the protocols are flexible. Firstly, adding a new node is easy: we need only to run the protocols once between the coordinator node and the new node. Secondly, removing a node is also easy, largely because it is convenient to re-run these protocols. The only two star protocol is SecPN V, which uses HBCs to directly transfer symmetric keys. When the user runs SecPN V, he/she should stay in a trusted place; thus we only give it two stars.

Resilience. The protocols with three stars use pair-wise keys; thus key leakage in one node will not influence the security links among other nodes, and will not influence the connectivity in the rest of the network. The protocols with two stars are slightly worse; the main problem is a single point of failure: if the coordinator is down, it will influence the security or connectivity in the rest of the network.

From the previous four metrics, we can see that protocols using pair-wise keys get more stars than other protocols. However, they not only consume more memory, but also bring more asymmetric computations in slave nodes. In Table 9.6, we list AsyS as the amount of asymmetric computations in all slave nodes, and AsyC as the amount of asymmetric computations in the coordinator. Since coordinators are always more powerful than slave nodes, the main metric here is AsyS. As we can see from this table, all the protocols using pair-wise keys bring more expensive computations in slave nodes. SecPN V is the most computation-efficient protocol.

The cryptography algorithms are beyond our concern, but their availability is still meaningful. One publicly available software package is TinyECC [70]. It realized a digital signature scheme (ECDSA), a key exchange protocol (ECDH), and a public key encryption scheme (ECIES). In addition, TinyPK, which is a RSA package, is also realized and reported in [115]. Besides, the energy consumption of ECC and RSA, especially low exponent

Table 9.6: Numbers of asymmetric computations

	AsyS	AsyC
HISP-K I	$(gs-1)(gs-1)$	$gs-1$
HISP-K II	$(gs-1)(gs-1)$	$gs-1$
HVSP-K I	$gs-1$	$gs-1$
HVSP-K II	$gs-1$	$gs-1$
HVSP-K III	$gs-1$	$gs-1$
HVSP-K IV	$gs-1$	$gs-1$
SecPN I	$gs-1$	$gs-1$
SecPN II	$gs-1$	$gs-1$
SecPN III	$(gs-1)(gs-1)$	$gs-1$
SecPN IV	$gs-1$	$gs-1$
SecPN V	0	0
HBSP-HW I	$4(gs-1)$	$4(gs-1)$
HBSP-HW II	$4(gs-1)(gs-1)$	$4(gs-1)$
HBSP-HW III	$4(gs-1)(gs-1)$	$4(gs-1)$

RSA, is acceptable. For example, it is reported that the energy consumption due to ECDH computation using Tmote Sky node is roughly 17mJ [70], and RSA verification using TelosB node is 2.7 mJ [90].

In summary, if nodes have enough computation capability and memory, HISP K I, HISP K II, SecPN III, HBSP-HW II, and HBSP-HW III are good choices. HVSP-K I, HVSP-K II, SecPN I, SecPN IV, and HBSP-HW I are good choices if star topology is good enough. Otherwise, HVSP-K III, HVSP-K IV, and SecPN II are good; they have average performance on most metrics. SecPN V is a good solution for implants, however, a trustworthy place is required.

9.3 Out-of-Band Channel Performance Review

Out-of-band channel performance is mainly related to channel speed, topological characteristics, and processing capabilities. Here, we only roughly categorize the channel speed as high speed and low speed. In addition, we can divide the channels based on topological characteristics: one way, two-way, one-to-one, multiple-to-one, one-to-multiple, and multiple-to-multiple. Besides, processing capabilities refers to how many different messages of particular sizes can be processed by the nodes in a specified time; the human user and computing devices have significantly different capabilities from the standpoint of processing

capabilities.

Human-controlled channels. From our daily experience, human-controlled channels should be modelled as two-way multiple-to-one channels with low speed and low processing capabilities in general. Human users are not good at sending or receiving a large amount of different messages in a limited time; thus these channels are channels with *low speed and low processing capabilities*. In addition, these channels are *two-way channels*: the user can receive information from nodes using displays, LEDs and buzzers, and can also send information to nodes using buttons. Besides, human users can compare messages that are transmitted using limited numbers of displays, LEDs and buzzers in most cases; thus human-controlled channels are *limited multiple-to-one channels*.

HVLC. HVLCs include HLSs and HLCs. HLSs have the following characteristics: transmission rate could be *high*; the processing capabilities of coordinator nodes could be *high*; and in many practical cases, they are *one-to-multiple* channels, which means that one LED can send signals to the light sensors of many nodes at the same time. HLCs have the following characteristics: the speeds of many HLCs are *low* due to the low video frame rate of cameras; the processing capabilities of coordinator nodes could be *high*; and in many practical cases, they are *multiple-to-one* channels, which means that cameras can collect LED data from many nodes at the same time.

HBC. Firstly, the speed is *fast* enough to transmit public keys, hash outputs, or digest outputs. Secondly, HBCs are *two-way* channels: two nodes attached to the same human body can exchange information. Thirdly, the processing capabilities of node could be *high*. Finally, HBCs are naturally one-to-multiple channels. Multiple receivers attached to the same human body can receive the electric signals from one transmitter at the same time.

9.4 Security Discussion

In this section, we give general comments on the security of our protocols, especially some easily ignored techniques or problems. Many attacks that will be discussed are related to the review of real-world attacks in Chapter 2.

Firstly, group size check is a significant process. It can help us to eliminate many attacks.

- Sybil attacks. In sybil attacks, one malicious node A illegitimately uses multiple identities. In our protocols, the *group size* is controlled by the human user, thus sybil attack is eliminated.

- Hello flooding. In Hello flooding attacks, A broadcasts "Hello" packets with high transmission power, thus other nodes think that A is their neighbour. In our protocols, the *group size* and *legal nodes* are controlled by the human user, thus users can prevent A from joining in or compromising a legal node in order to broadcast the "Hello" packets.

- Holes. In wormhole attacks, A tunnels messages received in one part of the network and replays them in another part. In sinkhole attacks, A manipulates the neighbouring nodes to lure nearly all the traffic from a particular area through a compromised node thereby creating a sink. In black hole attacks [10], A drops all the received packets. In selective forwarding attacks, A refuses to forward certain messages and simply drops them, ensuring that they are not propagated toward their destination. In each attack, either the *group size* is not correct, or the *hash/digest outputs* are not the same; thus these attacks are eliminated.

In addition, button pushing also plays an important role in DoS attack elimination. DoS attacks that aim to block the wireless radio channels can be easily detected: users find that the protocols *cannot be successfully finished*. However, DoS attacks that aim to *deplete battery life can succeed*, if one node does not require an access control mechanism, for example button pushing, to initiate a protocol run. Thus, in practical scenarios, such access control mechanisms should be always added in each node.

Besides, nonces are useful. For example, in time synchronization attacks, A deceives some nodes into thinking that an incorrect time is accurate. Since our protocols only rely on *nonces*, these attacks against time synchronization protocols will not influence it.

Furthermore, our protocols do not rely on a pre-distribution phase, and use asymmetric cryptography, thus passive attackers cannot get the security credentials unless nodes are compromised. However, these protocols *cannot prevent privacy information leakage*, e.g. identity leakage, and traffic analysis.

Finally, HCBK and SHCBK, which are the general versions of many protocols in this thesis, are verified under the Dolev-Yao channel model (for wireless radio channels) and NSB channel model (for out-of-band channels) using FDR, and no attacks have been found [27]. Meanwhile, we have also verified the general version of hash-based protocols, and no attacks have been found. However, we must emphasise that the *actions* mentioned in the protocol explanations are very important. One example is shown in Protocol 9.9. It is a two-party protocol that human users are not concerned about the signals on S in step 3. As we can see, although the message flow is not changed, attacks on S can succeed as in Attack 9.1.

Protocol 9.9 (Modified PH)**.**

1. $\mathsf{C} \longrightarrow \mathsf{S} : \mathsf{C}, Info_{\mathsf{C}}$
2. $\mathsf{S} \longrightarrow \mathsf{C} : \mathsf{S}, Info_{\mathsf{S}}$
3. $\mathsf{C} \Longrightarrow \mathsf{S} : H$

Attack 9.1 (One attack against Modified PH)**.**

1. $\mathsf{C} \longrightarrow \mathsf{A} : \mathsf{C}, Info_{\mathsf{C}}$

1. $\mathsf{A} \longrightarrow \mathsf{S} : \mathsf{C}, Info_{\mathsf{A}}$
2. $\mathsf{S} \longrightarrow \mathsf{C} : \mathsf{S}, Info_{\mathsf{S}}$
3. $\mathsf{C} \Longrightarrow \mathsf{S} : H$

Note that channel models that we used in this dissertation is NSB, DY, and CON [34, 31]; their meaning can be found in Table 3.2. The DY model is used to model wireless radio channels. The NSB model is used to model human-controlled channels in Chapter 3 and 4, human controlled visible light communication channels in Chapter 5 and 6, and human-controlled intra-body communication channels in Chapter 7 and 8. The CON model is used to model human-controlled intra-body communication channels in a trustworthy and isolated place, which is used in Chapter 7.

9.5 Chapter Summary

In this chapter, we have evaluated multi-channel security protocols mentioned in the thesis. The evaluations mainly focus on efficiency and security.

We have evaluated the protocols using metrics based on communications and key schemes. As our solution, key transport protocols such as HVSP-K III, IV, and SecPN II achieve a good balance; thus they should be firstly considered in general personal networks. However, if we want to establish security links to implants, SecPN V is the best protocol. Besides, if the nodes are relatively powerful, and the network scale is very small, we can also use for example HISP-K I, II, SecPN III, HBSP-HW II and III for better security.

Performance of out-of-band channels are reviewed. HBCs and HVLCs are fast, and can handle parallel different messages; and human-controlled channels are slow, and can only handle very limited parallel different messages. Multi-channel security protocols must be designed according to these characteristics.

Regarding security, we have emphasised the importance of group size control, button pushing, and failure signals. These may seem trivial, and might be easily ignored by users and developers, but they are actually very important to our protocols.

Chapter 10

Conclusion and Future Work

In this thesis, we have studied multi-channel security protocols in personal networks. With the help of out-of-band channels, especially NSB out-of-band channels, these protocols can bootstrap security in personal networks. In particular, three kinds of security protocol have been studied: HISPs that use human-controlled channels, HVSPs that use HVLCs, and HBSPs that use HBCs. Interesting trade-offs have been discovered among communication, computation and security, resulting from different channel implementations and protocols.

10.1 Conclusion

10.1.1 Human Interactive Security Protocols

Human Interactive Security Protocols (HISPs) in personal networks are one of our major contributions. HISPs are flexible. They *do not need pre-distributed secrets* to bootstrap security in wearable personal networks. Whenever the existing security credentials are compromised, users can *easily update* them using HISPs. We have studied these protocols from three aspects: human-controlled channels, security protocols using these channels, and use cases. They are described below.

Firstly, we have investigated available human-controlled channels in personal networks. Many of them are low-speed NSB out-of-band channels. Their security and networking properties are the foundations of HISPs.

Secondly, we have studied HISPs, which can bootstrap security in personal networks

with the help of human-controlled NSB channels. These protocols use keyed digest function; the digest output is short, so it can be easily transmitted over low-speed channels. However, digest function is vulnerable to combinatorial attacks. In HISPs, these attacks are eliminated using group commitment schemes. Altogether, HISPs are designed in order to achieve a balance between security and efficiency in personal networks.

Thirdly, two HISP use cases were studied. As the first use case, we have designed an ECDH version of HISPs, which provides a possible way of dynamically allocating and updating identities and ECDH public tokens. These protocols eliminate the primary security problem of ECDH: man-in-the-middle attacks. As the second use case, we designed several binding protocols, which can be used instead of key distribution protocols in some scenarios. These binding protocols can eliminate Sybil attacks. With these use cases, we show that HISPs are very useful; this new authentication philosophy can help us to solve some real-world problems in personal networks.

10.1.2 Security Protocols Using Visible Light Communications: HVSPs

HVSPs in personal networks are designed to *relieve the tedium of the digest comparison using LEDs*. Although cameras and image processing techniques are widely used as authentication tools today, we still have some interesting findings. Our contributions are described below.

Firstly, HVLCs were studied in personal networks. Except for the human-controlled LED-camera channels (HLCs), we found another kind of practical channel: human-controlled LED-sensor channels (HLSs), which use light sensors as LED signal receivers. In many cases, HVLCs are NSB channels; one major reason is that human users can see that the data comes from the desired authentic source. We have also analyzed the networking properties of HVLCs; it turns out that normal HLCs are low-speed channels, but HLSs can be fast.

Secondly, HVSPs have been proposed. These protocols reduce the burden of human users with the help of HVLCs; human users no longer need to compare digests. Additionally, we further optimize the protocols based on the networking properties of HVLCs: since cameras can read different LED signals at the same time, parallel protocols are a good choice; meanwhile, since HLSs can be fast, protocols that transmit many digests in a short

period is an acceptable solution.

Thirdly, two HVSP use cases are given. As the first use case, HVSPs have been used for key establishment. They reduce the burden of human users using HVLCs. In addition, the computation burden of slave nodes is also reduced. As the second use case, a secure positioning service system has been designed. Attacks against the localization system have been eliminated using HVSPs. These two use cases demonstrate that HVSPs are good security building blocks in personal networks, and potentially have wide usages.

10.1.3 Security Protocols Using Intra-Body Communications: HBSPs

HBSPs were studied because HISPs and HVSPs are not suitable for personal networks with *implants*. It is a unique experience that personal networks are securely bootstrapped using electric signals transmitted via the human body. Our experiences and contributions are described below.

Firstly, HBCs have been studied in personal networks. They use human tissue as the media of electric signals. These channels can provide high transmission speed using standard ports on nodes. In addition, our output voltage is only 3.3V, which is a safe voltage for most people. Most importantly, they are NSB channels in many cases: attacks such as spoofing or blocking signals transmitted inside the use's body can be eliminated.

Secondly, HBSPs have been proposed. These protocols can bootstrap security in personal networks with wearable or implanted nodes. Data can be directly exchanged using HBCs. Alternatively, HBCs can be used for verifying data exchanged in wireless radio channels. These protocols can be straightforward, which means that they can be easily understood and implemented.

Finally, two use cases have been provided. The first is SecPN, which uses HBCs in order to bootstrap a wearable personal network securely. The second is multi-channel identity-based protocols, which are able to eliminate attacks found in [100] against an identity-based protocol proposed in [48]. These use cases clearly show that HBSPs are promising security mechanisms in future wearable and implanted computing systems.

10.2 Future Work

10.2.1 Out-of-Band Channels

Usability The usability of out-of-band channels is one of the meaningful future directions. Many new out-of-band channels including HVLCs and HBCs have been proposed. However, the usability of these channels has not been thoroughly examined. It will be interesting to investigate their security and usability given different scenarios and modes of authentication. Based on this, we can improve the usability of existing or new out-of-band channels while achieving acceptable security.

Extending current solutions Although many out-of-band channels and their corresponding protocols have been studied, there are still much work that is interesting. For example, in the HBC based security system for implants, how can we design proper mechanism corresponding to different context (i.e. in everyday life and emergency scenario) is still an open issue.

New out-of-band channels There are many other potential out-of-band channels besides human controlled channels, HVLCs and HBCs. These out-of-band channels can be established using sound (the human voice), vibration, and acceleration. It will be interesting to explore these new out-of-band channels: the enabling techniques; their security and networking properties; and security protocols based on these channels.

10.2.2 Nano/Micro Communication Networks

Concepts Nano/micro communication networks are networks of nano/micro computing devices that are typically a few hundred nanometers or micrometers. Nano/micro scale computing devices have already been designed. Examples are a carbon nanotube sensor that changes its fluorescence after exposure to the glucose [32], a nano scale optical biosensor for Alzheimers disease [75], and ultra-small particles of iron oxide for pathologic tissue characterisation [76]. However, communication techniques are challenging in such small devices [32]. Nowadays, electromagnetic communication and molecular communication are

the two main approaches.

These networks are supposed to flow inside the human body. They can diffuse, for example, in the blood vessels and spinal canal in order to sense acute disease processes and monitor chronic illnesses.

Security Fundamental security requirements of these networks are similar to personal networks, especially when electromagnetic communication is used. Confidentiality is a legal requirement when these networks are used in medical applications. In addition, authenticity and integrity are always necessary in order to prevent data compromise.

It is difficult to deploy current cryptography-based mechanisms in these devices. These devices are extremely resource-constraint. There is not enough computation capability and storage for even symmetric cryptography algorithms.

Multi-channel security protocols can be one potential solution. For example, visible light communications and intra-body communications can be used for exchanging secrets when they are confidential channels. In addition, supposing the nano/micro devices exchange secrets only when they receive certain signals from out-of-band NSB channels; we will have more confidence regarding security.

10.2.3 Quantum Key Distribution

Quantum effects cannot be avoided when electronic devices become smaller and smaller. In addition, quantum communications offer some advantages over classic communications. Thus, in the near future, if quantum computing becomes available, we can provide security services using quantum key distributions.

Quantum key distributions are key establishment protocols using quantum communications. The first is BB84 developed by Charles Bennett and Gilles Brassard in 1984. Using this protocol, two parties Alice and Bob could negotiate a shared key via quantum channels. In the following years, many other protocols have also been proposed, for example, E91, SARG04, and KMB09. All use the characteristics of quantum communications in order to securely establish keys.

These characteristics can be roughly separated into two main categories. The first

category mainly relates to quantum indeterminacy of quantum states measurements. The second is entanglement; this means that for example measuring one object will affect the other.

In our case, we can use quantum key distributions instead of key establishment protocols using asymmetric cryptography. In this case, expensive computations are not required in small nodes. Alternatively, we can use quantum channels as out-of-band channels. However, the challenges are communication-enabling technologies, especially small and cheap equipment.

Appendix A

List of Acronyms

ACL Access Control List

AF Application Framework

AP Access Point

APS Application Support Sub-Layer

ATM Automated Teller Machine

BLE Bluetooth Low Energy

CON Confidential Channel, see Subsection 3.2.2

CRC Cyclic Redundancy Check

DAM Deployment Adjacent Matrix

DH Diffi-Hellman

DoS Denial of Service

DY Dolev-Yao Channel, see Subsection 3.2.2

ECC Elliptic Curve Cryptography

ECDH Elliptic Curve Diffie-Hellman Protocol

ECDSA Elliptic Curve Digital Signature Algorithm

ECG Electrocardiography

ECIES Elliptic Curve Integrated Encryption Scheme

EEG Electroencephalography

EEPROM Electrically Erasable Programmable Read Only Memory

E-G Eschenauer-Gligor Model

EMG Electromyography

FFD Full-Function Device

GPS Global Positioning System

GSM Global System for Mobile Communications

HBC Human-Controlled IBC, see Section 7.2

HBSP HB for HBC, SP for Security Protocol, see Section 7.3

HCBK Hash Commitment Before Knowledge

HCC Human-Controlled Channel, see Section 3.2

HISP Human Interactive Security Protocol, see Section 3.3

HLC H for human-controlled, LC for LED-camera channel, see Subsection 5.4.1

HLS H for human-controlled, LS for LED-sensor channel, see Subsection 5.4.1

HVLC H for human-controlled, VLC for visible light communication, see Subsection 5.4.1

HVSP H for human-controlled, V for visible light communication, SP for security protocol, see Subsection 5.5.1

HW Hölbl and Welzer Protocol

IBC Intra-Body Communications, see Section 7.1

ICMPv6 Internet Control Message Protocol version 6

IEEE Institute of Electrical and Electronics Engineers

IPsec Internet Protocol Security

IPv6 Internet Protocol version 6

LC LED-camera channel, see Section 5.3

LED Light-emitting diode

LLT Location Look Up Table

LN Localization Node

LS LED-sensor channel, see Section 5.2

MAC Media Access Control

MITM Man in the Middle

NB No-blocking Channel, see Subsection 3.2.2

NS No-spoofing Channel, see Subsection 3.2.2

NSB No-spoofing NB Channel, see Subsection 3.2.2

NWK Network Layer

OOB Out of Band

PHY Physical Layer

PKI Public Key Infrastructure

PN Personal Network

QoS Quality of Service

RAM Random Access Memory

RFD Reduced-Function Device

RGB Red Green Blue colour model

ROM Read Only Memory

SEC Secure Channel, see Subsection 3.2.2

SHCBK Symmetric HCBK

TLS Transport Layer Security

UDP User Datagram Protocol

VLC Visible Light Communication, see Subsection 5.1.1

WLAN Wireless Local Area Network

WPS Wi-Fi Positioning System

ZDO ZigBee Device Object

ZDP ZigBee Device Profile

ZSEC ZigBee Security Service

6LoWPAN IPv6 over Low power Wireless Personal Area Networks

References

[1] 6lowpan. http://datatracker.ietf.org/wg/6lowpan/.

[2] adidas micoach heart rate monitor. http://store.apple.com/uk/product/H8720Z/A/adidas-micoach-heart-rate-monitor.

[3] Bluetooth specifications. https://www.bluetooth.org/en-us/specification.

[4] Controls for wii. http://www.nintendo.com/wii/what-is-wii/#/controls.

[5] Nike+ipod. http://www.apple.com/ipod/nike/.

[6] Skyhook, inc. http://www.skyhookwireless.com.

[7] Zigbee specifications. http://www.zigbee.org/Specifications.aspx.

[8] *IEEE Standard for Local and metropolitan area networksPart 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs).* IEEE Press, New York, NY, USA, 2011.

[9] F. Adelstein, S. K. S. Gupta, G. Richard, and L. Schwiebert. *Fundamentals of mobile and pervasive computing.* McGraw-Hill, 2005.

[10] Nadeem Ahmed, Salil S. Kanhere, and Sanjay Jha. The holes problem in wireless sensor networks: a survey. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(2):4–18, April 2005.

[11] Hani Alzaid, Ernest Foo, and Juan Gonzalez Nieto. Secure data aggregation in wireless sensor network: a survey. In *Proceedings of the sixth Australasian conference on Information security - Volume 81*, AISC '08, pages 93–105, Darlinghurst, Australia, Australia, 2008. Australian Computer Society, Inc.

[12] A. Bag and M.A. Bassiouni. Energy efficient thermal aware routing algorithms for embedded biomedical sensor networks. In *Mobile Adhoc and Sensor Systems (MASS), 2006 IEEE International Conference on*, pages 604–609, 2006.

[13] C.R. Baker, K. Armijo, S. Belka, M. Benhabib, V. Bhargava, N. Burkhart, A. Der Minassians, G. Dervisoglu, L. Gutnik, M.B. Haick, C. Ho, M. Koplow, J. Mangold, S. Robinson, M. Rosa, M. Schwartz, C. Sims, H. Stoffregen, A. Waterbury, E.S. Leland, T. Pering, and P.K. Wright. Wireless sensor networks for home health care. In *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on*, volume 2, pages 832–837, 2007.

[14] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Proceedings of the 9th Annual Network and Distributed System Security Symposium (NDSS)*, pages 7–19. Citeseer, 2002.

[15] R. Blom. An optimal class of symmetric key generation systems. In *Advances in Cryptology*, pages 335–338. Springer, 1985.

[16] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. In *Advances in cryptology–CRYPTO92*, pages 471–486. Springer, 1993.

[17] Colin Boyd and Anish Mathuria. *Protocols for authentication and key establishment.* Springer, 2003.

[18] B. Braem, B. Latre, I. Moerman, C. Blondia, and P. Demeester. The wireless autonomous spanning tree protocol for multihop wireless body area networks. In *Mobile and Ubiquitous Systems: Networking Services, 2006 Third Annual International Conference on*, pages 1–8, 2006.

[19] B. Braem, B. Latre, I. Moerman, C. Blondia, E. Reusens, W. Joseph, L. Martens, and P. Demeester. The need for cooperation and relaying in short-range high path loss sensor networks. In *Sensor Technologies and Applications, 2007. SensorComm 2007. International Conference on*, pages 566–571, 2007.

[20] M. Cagalj, S. Capkun, and J. P. Hubaux. Key agreement in peer-to-peer wireless networks. *Proceedings of the IEEE*, 94(2):467–478, 2006.

[21] S.A. Camtepe and B. Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. *Computer Security–ESORICS 2004*, pages 293–308, 2004.

[22] D.W. Carman, P.S. Kruus, and B.J. Matt. Constraints and approaches for distributed sensor network security. *NAI Labs Technical Report #00-010*, September 2000.

[23] Giovanni Cennini, Jeremie Arguel, Kaan Akşit, and Arno van Leest. Heart rate monitoring via remote photoplethysmography with motion artifacts reduction. *Opt. Express*, 18(5):4867–4875, Mar 2010.

[24] S. Chalasani and J.M. Conrad. A survey of energy harvesting sources for embedded systems. In *Southeastcon, 2008. IEEE*, pages 442–447, 2008.

[25] H. Chan and A. Perrig. Pike: Peer intermediaries for key establishment in sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 1, pages 524–535. IEEE, 2005.

[26] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. *Security and Privacy, IEEE Symposium on*, 0:197, 2003.

[27] Bangdao Chen, L. H. Nguyen, and A.W. Roscoe. Reverse authentication in financial transactions and identity management. *Mobile Networks and Applications*, pages 1–16, 2012.

[28] Min Chen, Sergio Gonzalez, Athanasios Vasilakos, Huasong Cao, and Victor CM Leung. Body area networks: A survey. *Mobile Networks and Applications*, 16(2):171–193, 2011.

[29] S. Cherukuri, K.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. In *Parallel Processing Workshops, 2003. Proceedings. 2003 International Conference on*, pages 432–439, 2003.

[30] James Russell Clarke Sr and Phyllis Maurer Clarke. Sleep detection and driver alert apparatus, 1997. US Patent 5,689,241.

[31] S. J. Creese, M. H. Goldsmith, R. Harrison, A. W. Roscoe, P. Whittaker, and I. Zakiuddin. Exploiting empirical engagement in authentication protocol design. *Security in Pervasive Computing*, pages 119–133, 2005.

[32] João P Silva Cunha, Bernardo Cunha, António Sousa Pereira, William Xavier, Nuno Ferreira, and Luis Meireles. Vital-jacket: A wearable wireless vital signs monitor for patients' mobility in cardiology and sports. In *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2010 4th International Conference on*, pages 1–2. IEEE, 2010.

[33] Dorothy W. Curtis, Esteban J. Pino, Jacob M. Bailey, Eugene I. Shih, Jason Waterman, Staal A. Vinterbo, Thomas O. Stair, John V. Guttag, Robert A. Greenes, and Lucila Ohno-Machado. Smartan integrated wireless system for monitoring unattended patients. *Journal of the American Medical Informatics Association*, 15(1):44 – 53, 2008.

[34] D. Dolev and A. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, 1983.

[35] J Dong and H Zhu. Mobile ecg detector through gprs/internet. In *Computer-Based Medical Systems, 2004. CBMS 2004. Proceedings. 17th IEEE Symposium on*, pages 485–489. IEEE, 2004.

[36] W. Du, J. Deng, Y.S. Han, S. Chen, and P.K. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 1. IEEE, 2004.

[37] 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 Transactions on Information and System Security (TISSEC)*, 8(2):228–258, 2005.

[38] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462. IEEE, 2004.

[39] MagySeif El-Nasr and Thanos Vasilakos. Digitalbeing: An ambient intelligence interactive dance experience. In RaymondS.T. Lee and Vincenzo Loia, editors, *Computational Intelligence for Agent-based Systems*, volume 72 of *Studies in Computational Intelligence*, pages 233–263. Springer Berlin Heidelberg, 2007.

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

[41] Tia Gao, D. Greenspan, M. Welsh, R. Juang, and A. Alm. Vital signs monitoring and patient tracking over a wireless network. In *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, pages 102–105, 2005.

[42] M. Garbey, Nanfei Sun, A. Merla, and I. Pavlidis. Contact-free measurement of cardiac pulse based on the analysis of thermal imagery. *Biomedical Engineering, IEEE Transactions on*, 54(8):1418–1426, 2007.

[43] C. Gehrmann and K. Nyberg. Security in personal area networks. *Security for Mobility*, pages 191–230, 2004.

[44] Hassan Ghasemzadeh, Vitali Loseu, Eric Guenterberg, and Roozbeh Jafari. Sport training using body sensor networks: a statistical approach to measure wrist rotation for golf swing. In *Proceedings of the Fourth International Conference on Body Area Networks*, page 2. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

[45] Hassan Ghasemzadeh, Vitali Loseu, and Roozbeh Jafari. Wearable coach for sport training: A quantitative model to evaluate wrist-rotation in golf. *Journal of Ambient Intelligence and Smart Environments*, 1(2):173–184, 2009.

[46] Zabih Ghassemlooy, Wasiu Popoola, and Sujan Rajbhandari. *Optical wireless communications: system and channel modelling with MATLAB*. CRC Press, 2012.

[47] Rafael C Gonzalez, Richard E Woods, and Steven L Eddins. *Digital image processing using MATLAB*, volume 2. Gatesmark Publishing Tennessee, 2009.

[48] Marko Hölbl and Tatjana Welzer. Two improved two-party identity-based authenticated key agreement protocols. *Computer Standards and Interfaces*, 31(6):1056 – 1060, 2009.

[49] D. Huang, M. Mehta, D. Medhi, and L. Harn. Location-aware key management scheme for wireless sensor networks. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 29–42. ACM, 2004.

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

[51] Xin Huang, Rong Fu, Bangdao Chen, Tingting Zhang, and AW Roscoe. User interactive internet of things privacy preserved access control. In *Internet Technology And Secured Transactions, 2012 International Conferece For*, pages 597–602. IEEE, 2012.

[52] Xin Huang, Yang Jiang, Zuguang Liu, Theo Kanter, and Tingting Zhang. Privacy for mhealth presence. *International Journal of Next-Generation Networks (IJNGN)*, 2(4):33–44, 2010.

[53] P. Iso-Ketola, T. Karinsalo, and J. Vanhala. Hipguard: A wearable measurement system for patients recovering from a hip operation. In *Pervasive Computing Technologies for Healthcare, 2008. PervasiveHealth 2008. Second International Conference on*, pages 196–199, 2008.

[54] Suman Jana, Sriram Nandha Premnath, Mike Clark, Sneha K. Kasera, Neal Patwari, and Srikanth V. Krishnamurthy. On the effectiveness of secret key extraction from wireless signal strength in real environments. In *Proceedings of the 15th annual*

international conference on Mobile computing and networking, MobiCom '09, pages 321–332, New York, NY, USA, 2009. ACM.

[55] Yungtaek Jang and M.M. Jovanovic. A contactless electrical energy transmission system for portable-telephone battery chargers. *Industrial Electronics, IEEE Transactions on*, 50(3):520–527, 2003.

[56] Shanshan Jin, So-Youn Park, and Ju-Jang Lee. Driver fatigue detection using a genetic algorithm. *Artificial Life and Robotics*, 11(1):87–90, 2007.

[57] Ronald Kainda, Ivan Flechais, and A. W. Roscoe. Usability and security of out-of-band channels in secure device pairing protocols. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, SOUPS '09, pages 11:1–11:12, New York, NY, USA, 2009. ACM.

[58] Ronald Kainda, Ivan Flechais, and A. W. Roscoe. Two heads are better than one: security and usability of device associations in group scenarios. In *Proceedings of the Sixth Symposium on Usable Privacy and Security*, SOUPS '10, pages 5:1–5:13, New York, NY, USA, 2010. ACM.

[59] H. Karl and A. Willig. *Protocols and architectures for wireless sensor networks*. Wiley-Interscience, 2007.

[60] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. *Ad Hoc Networks*, 1(23):293 – 315, 2003.

[61] A. Khisti and Gregory W. Wornell. Secure transmission with multiple antennas i: The misome wiretap channel. *Information Theory, IEEE Transactions on*, 56(7):3088–3104, 2010.

[62] A. Khisti and Gregory W. Wornell. Secure transmission with multiple antennas ii: The mimome wiretap channel. *Information Theory, IEEE Transactions on*, 56(11):5515–5532, 2010.

[63] Arun Kumar, Nitesh Saxena, Gene Tsudik, and Ersin Uzun. A comparative study of secure device pairing methods. *Pervasive and Mobile Computing*, 5(6):734 – 749, 2009.

[64] B. Latre, B. Braem, I. Moerman, C. Blondia, E. Reusens, W. Joseph, and P. Demeester. A low-delay protocol for multihop wireless body area networks. In *Mobile and Ubiquitous Systems: Networking Services, 2007. MobiQuitous 2007. Fourth Annual International Conference on*, pages 1–8, 2007.

[65] Benoît Latré, Bart Braem, Ingrid Moerman, Chris Blondia, and Piet Demeester. A survey on wireless body area networks. *Wireless Networks*, 17(1):1–18, 2011.

[66] J. Lee and D.R. Stinson. A combinatorial approach to key predistribution for distributed sensor networks. In *Wireless Communications and Networking Conference, 2005 IEEE*, volume 2, pages 1200–1205. IEEE, 2005.

[67] Philip Levis, Sam Madden, Joseph Polastre, Robert Szewczyk, Kamin Whitehouse, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, et al. Tinyos: An operating system for sensor networks. In *Ambient intelligence*, pages 115–148. Springer, 2005.

[68] Huaming Li and Jindong Tan. An ultra-low-power medium access control protocol for body sensor network. In *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, pages 2451–2454, 2005.

[69] Huaming Li and Jindong Tan. Heartbeat-driven medium-access control for body sensor networks. *Information Technology in Biomedicine, IEEE Transactions on*, 14(1):44–51, 2010.

[70] A. Liu and P. Ning. Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pages 245–256. IEEE Computer Society, 2008.

[71] Donggang Liu, Peng Ning, and Rongfang Li. Establishing pairwise keys in distributed sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8:41–77, February 2005.

[72] Tie Liu and S. Shamai. A note on the secrecy capacity of the multiple-antenna wiretap channel. *Information Theory, IEEE Transactions on*, 55(6):2547–2553, 2009.

[73] Joe Loughry and David A. Umphress. Information leakage from optical emanations. *ACM Trans. Inf. Syst. Secur.*, 5(3):262–289, August 2002.

[74] David Malan, Thaddeus Fulford-Jones, Matt Welsh, and Steve Moulton. Codeblue: An ad hoc sensor network infrastructure for emergency medical care. In *International workshop on wearable and implantable body sensor networks*, volume 5, 2004.

[75] Michael Manzo, Tanya Roosta, and Shankar Sastry. Time synchronization attacks in sensor networks. In *Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks*, SASN '05, pages 107–116, New York, NY, USA, 2005. ACM.

[76] Suhas Mathur, Wade Trappe, Narayan Mandayam, Chunxuan Ye, and Alex Reznik. Radio-telepathy: extracting a secret key from an unauthenticated wireless channel. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, MobiCom '08, pages 128–139, New York, NY, USA, 2008. ACM.

[77] K. Montgomery, C. Mundt, G. Thonier, A. Tellier, U. Udoh, V. Barker, R. Ricks, L. Giovangrandi, P. Davies, Y. Cagle, J. Swain, J. Hines, and G. Kovacs. Lifeguard - a personal physiological monitor for extreme environments. In *Engineering in Medicine and Biology Society, 2004. IEMBS '04. 26th Annual International Conference of the IEEE*, volume 1, pages 2192–2195, 2004.

[78] Anirudh Natarajan, Mehul Motani, Buddhika de Silva, Kok-Kiong Yap, and K. C. Chua. Investigating network architectures for body sensor networks. In *Proceedings of the 1st ACM SIGMOBILE international workshop on Systems and networking support for healthcare and assisted living environments*, HealthNet '07, pages 19–24, New York, NY, USA, 2007. ACM.

[79] James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. The sybil attack in sensor networks: analysis & defenses. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, IPSN '04, pages 259–268, New York, NY, USA, 2004. ACM.

[80] Jason WP Ng, Benny PL Lo, Oliver Wells, Morris Sloman, Nick Peters, Ara Darzi, Chris Toumazou, and Guang-Zhong Yang. Ubiquitous monitoring environment for wearable and implantable sensors (ubimon). In *International Conference on Ubiquitous Computing (Ubicomp)*, 2004.

[81] L. H. Nguyen and A. W. Roscoe. Authenticating ad hoc networks by comparison of short digests. *Information and Computation*, 206(2-4):250–271, 2008.

[82] L. H. Nguyen and A. W. Roscoe. Authentication protocols based on low-bandwidth unspoofable channels: a comparative survey. *Journal of Computer Security*, 19(1):139–201, 2011.

[83] F. Oggier and B. Hassibi. The secrecy capacity of the mimo wiretap channel. *Information Theory, IEEE Transactions on*, 57(8):4961–4972, 2011.

[84] Katsuyuki Okeya and Tetsu Iwata. Side channel attacks on message authentication codes. In *Security and Privacy in Ad-hoc and Sensor Networks*, pages 205–217. Springer, 2005.

[85] N. Oliver and F. Flores-Mangas. Healthgear: a real-time wearable system for monitoring and analyzing physiological signals. In *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*, pages 4 pp.–64, 2006.

[86] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 641–648. IEEE, 2006.

[87] B. Otal, L. Alonso, and C. Verikoukis. Highly reliable energy-saving mac for wireless body sensor networks in healthcare systems. *Selected Areas in Communications, IEEE Journal on*, 27(4):553–565, 2009.

[88] Julien Pansiot, Benny Lo, and Guang-Zhong Yang. Swimming stroke kinematic analysis with bsn. In *Body Sensor Networks (BSN), 2010 International Conference on*, pages 153–158. IEEE, 2010.

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

[90] Krzysztof Piotrowski, Peter Langendoerfer, and Steffen Peter. How public key cryptography influences wireless sensor node lifetime. In *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, pages 169–176. ACM, 2006.

[91] Ming-Zher Poh, Daniel J. McDuff, and Rosalind W. Picard. Non-contact, automated cardiac pulse measurements using video imaging and blind source separation. *Opt. Express*, 18(10):10762–10774, May 2010.

[92] A. Purwar, Do-Un Jeong, and Wan-Young Chung. Activity monitoring from real-time triaxial accelerometer data using sensor network. In *Control, Automation and Systems, 2007. ICCAS '07. International Conference on*, pages 2402–2406, 2007.

[93] Hongliang Ren and M.Q.-H. Meng. Rate control to reduce bioeffects in wireless biomedical sensor networks. In *Mobile and Ubiquitous Systems: Networking Services, 2006 Third Annual International Conference on*, pages 1–7, 2006.

[94] Tanya Roosta, Shiuhpyng Shieh, and Shankar Sastry. Taxonomy of security attacks in sensor networks and countermeasures. In *The First IEEE International Conference on System Integration and Reliability Improvements*, volume 25, page 94, 2006.

[95] A. G. Ruzzelli, R. Jurdak, G. M.P O'Hare, and P. Van Der Stok. Energy-efficient multi-hop medical sensor networking. In *Proceedings of the 1st ACM SIGMOBILE international workshop on Systems and networking support for healthcare and assisted living environments*, HealthNet '07, pages 37–42, New York, NY, USA, 2007. ACM.

[96] D.S. Sanchez and H. Baldus. A deterministic pairwise key pre-distribution scheme for mobile sensor networks. In *Security and Privacy for Emerging Areas in Communi-*

cations Networks, 2005. SecureComm 2005. First International Conference on, pages 277–288. IEEE, 2005.

[97] A. Sathyanarayana, S. Nageswaren, H. Ghasemzadeh, R. Jafari, and J.H.L. Hansen. Body sensor networks for driver distraction identification. In *Vehicular Electronics and Safety, 2008. ICVES 2008. IEEE International Conference on*, pages 120–125, 2008.

[98] S. Shafiee, Nan Liu, and Sennur Ulukus. Towards the secrecy capacity of the gaussian mimo wire-tap channel: The 2-2-1 channel. *Information Theory, IEEE Transactions on*, 55(9):4033–4039, 2009.

[99] Claude E Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949.

[100] Kyung-Ah Shim. Cryptanalysis of two identity-based authenticated key agreement protocols. *Communications Letters, IEEE*, 16(4):554–556, 2012.

[101] Hang Su and Xi Zhang. Battery-dynamics driven tdma mac protocols for wireless body-area monitoring networks in healthcare applications. *Selected Areas in Communications, IEEE Journal on*, 27(4):424–434, 2009.

[102] Ali Maleki Tabar, Arezou Keshavarz, and Hamid Aghajan. Smart home care network using sensor fusion and distributed vision-based reasoning. In *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, VSSN '06, pages 145–154, New York, NY, USA, 2006. ACM.

[103] Daisuke Takahashi, Yang Xiao, Fei Hu, Jiming Chen, and Youxian Sun. Temperature-aware routing for telemedicine applications in embedded biomedical sensor networks. *EURASIP J. Wirel. Commun. Netw.*, 2008:26:1–26:26, January 2008.

[104] T. Taleb, D. Bottazzi, and N. Nasser. A novel middleware solution to improve ubiquitous healthcare systems aided by affective information. *Information Technology in Biomedicine, IEEE Transactions on*, 14(2):335–349, 2010.

[105] S Taranovich. Medical sensors encompass biomedical electronics. *EDN*, 56:35–42, 2011.

[106] Nils Ole Tippenhauer, Kasper Bonne Rasmussen, Christina Pöpper, and Srdjan Čapkun. Attacks on public wlan-based positioning systems. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, MobiSys '09, pages 29–40, New York, NY, USA, 2009. ACM.

[107] Sana Ullah, Henry Higgins, Bart Braem, Benoit Latre, Chris Blondia, Ingrid Moerman, Shahnaz Saleem, Ziaur Rahman, and Kyung Sup Kwak. A comprehensive survey of wireless body area networks. *Journal of Medical Systems*, 36(3):1065–1094, 2012.

[108] AT van Halteren, RGA Bults, KE Wac, Dimitri Konstantas, IA Widya, NT Dokovski, GT Koprinkov, VM Jones, and Rainer Herzog. Mobile patient monitoring: The mobihealth system. 2004.

[109] S. Vaudenay. Secure communications over insecure channels based on short authenticated strings. In *Advances in Cryptology–CRYPTO 2005*, pages 309–326. Springer, 2005.

[110] Serge Vaudenay. Secure communications over insecure channels based on short authenticated strings. In Victor Shoup, editor, *Advances in Cryptology CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 309–326. Springer Berlin Heidelberg, 2005.

[111] K.K. Venkatasubramanian, A. Banerjee, and S. K S Gupta. Plethysmogram-based secure inter-sensor communication in body area networks. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pages 1–7, 2008.

[112] K.K. Venkatasubramanian, Venkatasubramanian, A. Banerjee, and S.K.S. Gupta. Ekg-based key agreement in body sensor networks. In *INFOCOM Workshops 2008, IEEE*, pages 1–6, 2008.

[113] Krishna K. Venkatasubramanian and Sandeep K. S. Gupta. Physiological value-based efficient usable security solutions for body sensor networks. *ACM Trans. Sen. Netw.*, 6(4):31:1–31:36, July 2010.

[114] Weichao Wang and Bharat Bhargava. Visualization of wormholes in sensor networks. In *Proceedings of the 3rd ACM workshop on Wireless security*, pages 51–60. ACM, 2004.

[115] R. Watro, D. Kong, S. Cuti, C. Gardiner, C. Lynn, and P. Kruus. Tinypk: securing sensor networks with public key technology. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 59–64. ACM, 2004.

[116] Maarten Wegdam. Awareness: A project on context aware mobile networks and services. In *14th Mobile & Wireless Communication Summit*, 2005.

[117] Marc Simon Wegmller. Personal area network (pan). 1995.

[118] Marc Simon Wegmller. Intra-body communication for biomedical sensor networks. 2007.

[119] F. L. Wong and F. Stajano. Multichannel security protocols. *IEEE Pervasive Computing*, pages 31–39, 2007.

[120] A. Wood and J.A. Stankovic. Denial of service in sensor networks. *Computer*, 35(10):54–62, 2002.

[121] A Wood, G Virone, T Doan, Q Cao, L Selavo, Y Wu, L Fang, Z He, S Lin, and J Stankovic. Alarm-net: Wireless sensor networks for assisted-living and residential monitoring. *University of Virginia Computer Science Department Technical Report*, 2, 2006.

[122] Aaron D Wyner. The wire-tap channel. *Bell Syst. Tech. J.*, 54(8):1334–1387, 1975.

[123] Yanming Xiao, Jenshan Lin, O. Boric-Lubecke, and V.M. Lubecke. Frequency-tuning technique for remote detection of heartbeat and respiration using low-power double-

sideband transmission in the ka-band. *Microwave Theory and Techniques, IEEE Transactions on*, 54(5):2023–2032, 2006.

[124] T. Zasowski, F. Althaus, M. Stager, A. Wittneben, and G. Troster. Uwb for non-invasive wireless body area networks: channel measurements and results. In *Ultra Wideband Systems and Technologies, 2003 IEEE Conference on*, pages 285–289, 2003.

[125] Y. Zhang, W. Liu, W. Lou, and Y. Fang. Securing sensor networks with location-based keys. In *Wireless Communications and Networking Conference, 2005 IEEE*, volume 4, pages 1909–1914. IEEE, 2005.

[126] Y. Zhang, W. Liu, W. Lou, and Y. Fang. Location-based compromise-tolerant security mechanisms for wireless sensor networks. *Selected Areas in Communications, IEEE Journal on*, 24(2):247–260, 2006.

[127] Gang Zhou, Jian Lu, Chieh-Yih Wan, M.D. Yarvis, and J.A. Stankovic. Bodyqos: Adaptive and radio-agnostic qos for body sensor networks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 565–573, 2008.

[128] Yun Zhou, Yanchao Zhang, and Yuguang Fang. Access control in wireless sensor networks. *Ad Hoc Networks*, 5(1):3 – 13, 2007. Security Issues in Sensor and Ad Hoc Networks.

[129] Hongjie Zhu, Wai Chiu Ng, Hengying Shan, and Jie Yuan. A physical layer security analysis on the electric-field intra-body communication. In *Computing, Networking and Communications (ICNC), 2012 International Conference on*, pages 14–17, 2012.