A SystemC Simulator for Secure Data Transfer in Healthcare Internet of Things

UNIVERSITY OF TURKU
Department of Future Technologies
Master of Science in Technology Thesis
Networked Systems Security
June 2019
Naren Purighalla

Supervisors:
Dr. Seppo Virtanen
Dr. Jouni Isoaho

UNIVERSITY OF TURKU
Department of Future Technologies

NAREN PURIGHALLA: A SYSTEMC SIMULATOR FOR SECURE DATA TRANSFER IN HEALTHCARE INTERNET OF THINGS

Master of Science in Technology Thesis
Networked Systems Security
June 2019

In this thesis, a simulator for secure data transfer between medical sensors and end-users is developed using smart e-health gateways in medical environments like hospitals, healthcare centers and old-age homes. The rate of adoption of Internet of Things (IoT) and the associated technology is on the rise but the security measures that accompany the said technology and the devices is not totally dependable. And the repercussions get serious when dealing with medical sensors because people's lives are at stake.

With medical sensors, due to their resource and energy constraints, it is difficult to apply strong and heavy cryptographic techniques to achieve maximum security. The use of smart e-health gateways here eases the burden on sensors by authenticating and authorizing the end-users on behalf of the sensors. This is achieved using certificate-based DTLS handshake protocol between the gateways and the end-user. Then, for end-to-end secure communication, session resumption is carried out between the sensors and the end-users which is not as energy consuming as the handshake.

The whole simulator is designed using SystemC, which is a library of C++ classes. It is chosen for this implementation because of various advantages it has over other hardware programming languages. SystemC, along with its verification library covers almost all aspects related to system design and modeling and the syntax of SystemC and C++ are the same. All the components of the hardware design are defined as modules here and the target is to achieve communication between these modules.

Finally, we calculate the time it takes for a set of values to go from the sensor to the gateway and the gateway to the end-user.

Table of Contents

Chapter 1

Introduction

The usage of Internet of Things (IoT) has been on the rise since the turn of the century. There are examples of many wearables, homes, industries, manufacturing units and even some sections of cities which have adopted this concept of objects sharing data with each other over a network which eases the data flow by reducing the human element in the process.

Statistics from different credible sources reveal that the adoption rate of IoT is very high, especially in the industrial sectors. Louis Columbus' report [1], claims that by 2020, different industrial sectors like discrete manufacturing, transportation and logistics and utilities are expected to spend around $40 billion on IoT. Not just these, the Information and Communications Technology (ICT) sector is projected to spend around $581 billion. The differences between how much money was spent on IoT (and therefore, how much IoT's technologies were used) between 2015 and 2020 can be seen from the report.

Discrete manufacturing sector is projected to experience an increase of 300% with total expenditure around $40 billion in 2020 from $10 billion in 2015. Similar amount of expenditure is projected in the Transportation and Logistics sector. Utilities is expected to spend even more, almost around 500% from $7 billion in 2015 to $40 billion in 2020. The biggest change though is expected in business-to-consumer (B2C) from $5 billion in 2015 to $25 billion in 2020.

Other industrial sectors are also expected to have a significant growth in terms of spending. Manufacturing, Healthcare, Energy and natural resources, Retail, Government and Insurance sectors have also adopted IoT technologies in recent years.

The rise of IoT directly implies that the reliance on technology is high and this reliance leads to fewer errors that are related to the "human element" i.e. the errors or accidents that happen due to a person's actions. But it does not take away the vulnerabilities that come with the technology itself or in most cases how people use the said technology. Most of the vulnerabilities in these IoT devices are found due to the firmware used being not up to date because either the manufacturers delay in releasing the updates or the updates might be too large for the existing device.

The vulnerabilities in the devices often lead to attacks from cyber-criminals. According to Symantec's report [2], the services that were mostly attacked in 2017 were Telnet, HTTP and HTTPS. Since some of the devices are resource constrained, the security in those devices may be very weak. Distributed Denial of Service (DDoS) attacks are very prevalent when one generally talks about IoT attacks.

The most famous DDoS attack on IoT devices - the Mirai attack [3] took place in 2016 where the intention was to just make some money off people who play "Minecraft", but the lack in security was evident in most of the devices connected such that much of the internet was not accessible to people on the east coast of the United States.

Later, when the Mirai source code was published online, many other cyber-criminals developed this code further to take down other areas of the internet infrastructure [4]. This code infects those devices which use the default username and password by using telnet and then the code coordinates all these devices to carry out a DDoS attack against victims.

According to Kaspersky's analysis [5], the malware samples for IoT devices that they encountered in the last three years have been increasing exponentially. In the first half of 2018 alone, the malware samples brought to their notice were 121 thousand which was more than three times the amount in 2017 at 32 thousand. In 2016, the number of samples were only around 3 thousand.

The rate of increase of IoT devices and therefore the rate of increase of attacks on them makes it really important to secure these devices and to secure the communication between the devices and network they use. One of the prime applications of IoT where the lack of security can have bigger consequences is the Healthcare sector.

In a medical environment like a hospital, a patient may have a few medical sensors reading data from him/her continuously and this data is transferred to the hospital staff for careful monitoring of the patient. If this communication is not secure, there are many possibilities on how malicious users can take advantage. Cyber-criminals can read the confidential data that is transferred, they can also tamper the data which can have huge repercussions and alter the destination of the data. So, improving the healthcare sector by having a secure communication between medical sensors can not only ease the workflow of data transfer in medical environments, but can also prevent this data from being accessible to malicious users.

The simulation presented in this thesis is based on the original research carried out by Moosavi et al. [6][7] and it uses “Smart e-Health Gateways” which act as a bridging point between medical sensors in smart hospitals and the internet. These gateways’ main task is to collect data from a group of medical sensors and transfer them back over a network to the end-user. These gateways help with the patients’ mobility in these smart homes or hospitals where the situation normally for the patients is confined to a particular room. An additional advantage of these gateways is that they are stationary and hence the resources used up are minimal. The results of the research showed that power-consumption, memory and communication bandwidth are all optimal so that the smart gateways can actually take up a part of the workload of the medical sensors. The communication between the smart e-health gateways and the end-user is secured by the use of a certificate-based DTLS handshake protocol which is the main IP security solution for IoT.

The simulation for the thesis is done using SystemC which is a system design library of C++ classes. SystemC is useful in this scenario because it helps designers with real productivity gains by letting both the hardware and the software components be designed together at a higher level of abstraction and it gives the designers an early understanding in the design, the complications and the interactions between different functions of the design. SystemC can also be advantageous because of functional verification, use of test benches and supporting register-transfer-level (RTL) modeling and synthesis. Figure. 1 [8], shows the versatility of SystemC over other languages.

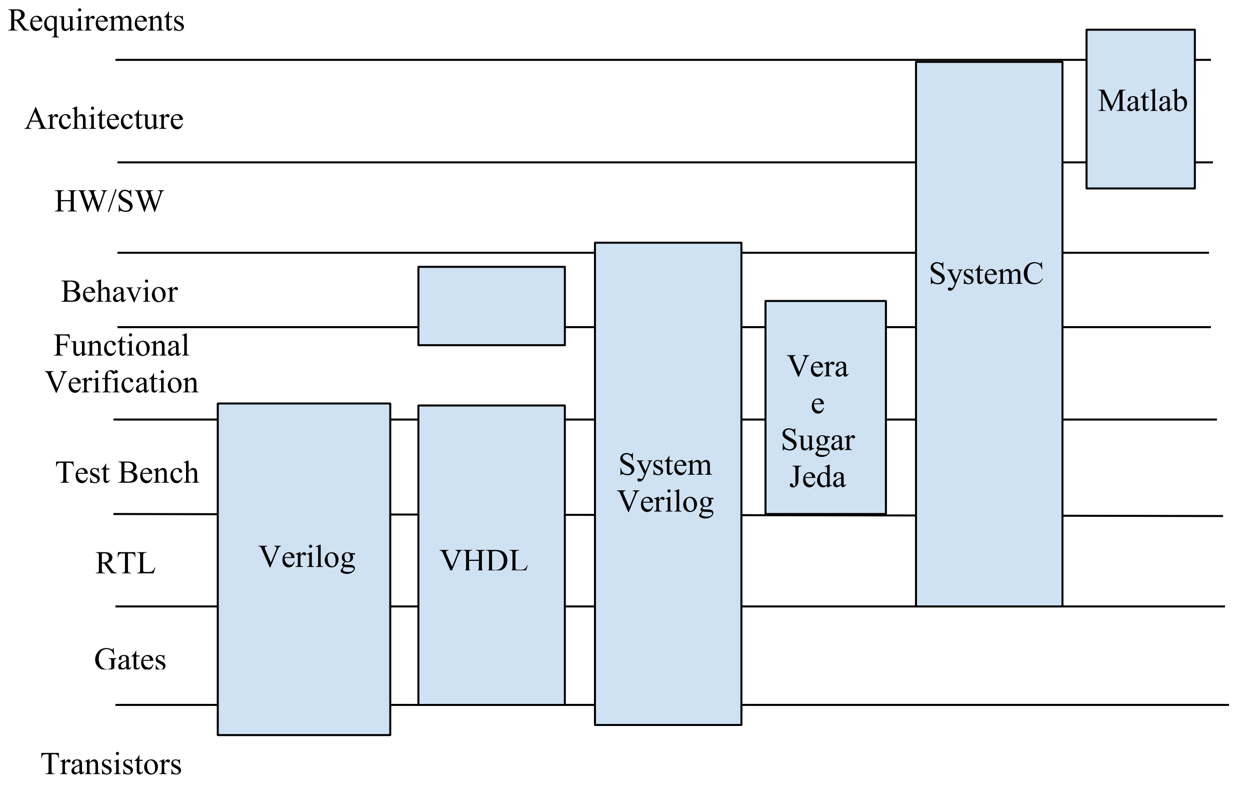

Figure 1. Comparison of Hardware Programming Languages

All these, and the syntax of SystemC which is the same as the syntax of C++ are the reasons for choosing the language.

With this simulator, it can be demonstrated how the data transfer takes place between medical sensors and the end-user. This data transfer can be broken into two parts. The first one where the data is read from a file and is transferred to the gateways using signals and ports in SystemC and the second part where the same data is transferred from the smart gateways to the end-user by securing it with DTLS handshake protocol.

The rest of the thesis is organized as follows. In Chapter 2, IoT is described in larger detail and why security matters in healthcare IoT. In Chapter 3, principles of SystemC are looked at along with the basics of compiling and running a program in Linux. In Chapter 4, the design aspect of this project is explained and how it is implemented in SystemC. In Chapter 5, the results and tests of the simulation are noted. Chapter 6 is the conclusion of the thesis with suggestions of improvements.

Chapter 2

Internet of Things

As discussed in the introduction, Internet of Things (IoT) is a network connection of devices which are able function and perform tasks by themselves without any human intervention. According to Li et al. [9], this concept was proposed by Kevin Ashton in 1999, where he explained IoT as a "uniquely identifiable interoperable connected objects with radio-frequency identification (RFID) technology".

The concept of IoT was first implemented by using RFID technology in logistics and pharmaceutical production. Later, with the rise of wireless sensing technologies, IoT then started using these and develop them into barcodes, intelligent sensing, cloud computing and low energy wireless communication. In the last few years, IoT has evolved to connecting physical things through a network like the internet and be able to access them. IoT is not in its nascent stages anymore with more and more organizations adopting this technology and upgrade their infrastructure, communication protocols and standards and interfaces to suit their needs. This reach of this technology is not just limited to private organizations. Many nations started to invest in IoT to not only meet their immediate needs but also develop this technology further to suit different applications. There are many sectors within urban places that are adopting this "smart city" feature which uses IoT technology to incorporate into their everyday lives.

2.1 Standards

Many international organizations like the International Telecommunication Union (ITU), Electronic Product Code global (EPCglobal), the International Electro-Technical Commission (IEC), the ISO, IEEE, the European Telecommunications Standards Institute (ETSI) and the American Standards National Institute (ANSI) have come up with a few standards for the usage of primary IoT technologies such as RFID and Wireless Sensor Networks (WSNs).

IEEE 802.15.4 for Zigbee, IEEE 802.11 for WLAN, IEEE 802.15.6 for wireless body area networks, IEEE 802.15.1 for Bluetooth, IPv6, 3G and 4G are some of the standards used for communication while ITU-T and IETF are the standards used for Quality of Service (QoS) whereas different ISO standards are used for RFID for different frequencies like ISO 18000-4 for 2.45 GHz, ISO 18000-6 for 860 to 960 MHz and ISO 18000-7 for 433 MHz.

These standards for IoT are important especially considering the rate of growth of IoT which is very high, and it can improve the rate at which IoT technologies are spread. These standards also help designers and developers to pick the best protocols that are used for various services in different applications in IoT.

2.2 Architecture

When different devices are connected to form a network, there has to be a functioning architecture which can connect the physical and digital worlds of these devices. When an architecture is designed, many factors like scalability, networking, interoperability of different kinds of devices, communication, business models and security have to be considered. To ensure the interoperability among different devices with different objectives, a service-oriented architecture (SoA) is necessary [9]. A generic SoA has four layers which interact with each other but each has different functions. These four layers are sensing layer, network layer, service layer and interface layer. Some other sources define IoT architecture in three layers with service layer and interface layer combining into one application layer.

The sensing layer is made up of devices with sensing capabilities, i.e. with sensors. Devices such as RFID tags, WSNs, Bluetooth devices make up the sensing layer. These devices will be able to exchange information with each other automatically. They use data sensing acquisition protocols to fetch data from their environments. When setting up the sensing layer, factors like communication, organization of these devices to form a network (like single-hop, multi-hop, mesh), resource constraints, energy consumption and sizes should be taken into consideration so the data collection is done seamlessly.

Network layer's functions are to connect the devices and take note of the environment that the devices are part of. Data transfer between the devices where data aggregation and decision making are then done. To have integrity of different functions that are used by the devices, Quality of Service (QoS) might be used by the network. Different devices which are present in the network are allotted roles to manage, deploy and time different tasks to be able to work smoothly and be able to collaborate between themselves. Things that need to be addressed in the network layer are security and privacy, QoS requirements, managing different wireless and mobile networks, different technologies for mining and data processing.

Service layer banks on the kind of technology which helps the IoT applications work steadily by relying on middleware technology. This technology can provide a platform wherein both the hardware and the software platforms can be reused. The services which are run on the network automatically try to explore new services for different applications and try to extract metadata.

The main tasks of the service layer are data exchange, data storage, data management, communication and search engines. Different components of the service layer where these activities take place are service discovery, service composition, trustworthiness management and service APIs. Service discovery finds different objects which can provide different services. Service composition is responsible for connecting different objects and maintain an interaction. Trustworthiness management processes the information gathered and service APIs provide communication between services as required by the consumers.

The interface layer addresses the issues of compatibility of different devices in the network which are manufactured by various vendors and lack a coherent set of standards. Differences may occur during information exchange and processing events. An interface profile (IFP) can be setup to provide information on various applications and services between devices.

2.3 Reference Architectures

Weyrich and Ebert [10] claim that the lack of standards for architecture for industrial internet and connectivity is a major threat in IoT and this need for reference architectures is felt with the number of initiatives that are being developed towards architecture standards. Reference architectures are helpful because they can be used as a guideline for other architectures. A reference architecture must be able to address requirements of connectivity, communication, device management, data collection, analysis, scalability and security features. A reference model is formed when details can be mentioned about how these entities interact with each other.

There are two major available architectures [10] - the Industrial Internet Reference Architecture (IIRA) and the Internet of Things - Architecture (IoT-A) where the IoT-A is described in detail.

The IoT-A, in terms of interpretation of data to extend it to business scenarios, focuses on generic informatics and covers almost the entirety of data management in cloud and servers and networking, transport and data links where machine-to-machine (M2M) communication is given focus rather than the Open Systems Interconnection (OSI) stack. The IoT-A is also based on sensors and tags. The main differences between IoT-A and the IIRA is that the IIRA centers around the functions of the industrial sector, its operations, the information transfer and the applications.

Other reference architectures are the Reference Architecture Model Industry 4.0 (RAMI 4.0) which started in Germany and is for smart factories for IoT standards, Arrowhead Framework which collaborates automation for embedded devices which are open networked and the Standard for an Architectural Framework for the Internet of Things (IoT) which centers on security, privacy, protection and safety issues.

2.4 Technology

For different devices to stay connected and have various sensors in them continuously monitor the surroundings while receiving and sending data, both the hardware and the software technologies must be used wholly. Whitmore et al. [11], classify the technology into hardware, software and architecture.

Most of the hardware infrastructure that is used in IoT networks is for sensor networks, Radio-Frequency Identification (RFID) and Near Field Communication (NFC).

RFID communication works in a way where the RFID reader recognizes the RFID tag placed next to it by radio-frequency electromagnetic fields. IoT applications mostly use RFID tags to send information in the form of Electronic Product Code (EPC) which is a form of UUID. Although the use of RFIDs has started even before the advancements of IoT, RFID's technology has an important place in different applications like logistics and supply chain management, food safety, retail industry and aviation. This technology can be used to make the data exchange over a network.

NFC is a technology that is built on RFID. In this communication standard, devices are able to exchange information over a short distance or when they are in contact with each other. Similar to RFID, each NFC tag has a UID which is connected to the tag. Extension of NFC is the Bluetooth technology which can be used to send and receive information between two mobile devices which are in close proximity.

Sensors are used to observe the surroundings for some data like temperature and humidity. A sensor network is formed when a group of sensors are used together for one common purpose and interact with each other. A wireless sensor network is formed when the network is wireless and may contain gateways which act as a bridge between the sensors and the network. Sensors just collect data from the environment they are placed in. To affect the environment they are placed in, actuators are used. They are used to transmit sound, light and even smells. They can be used along with sensors to form actuator-sensor network. An example of this network is a fire alarm, where a sensor is used to detect smoke or the fire and the actuator produces loud noise to warn people about the fire.

The main difference between hardware and software architecture is that IoT needs new software on a regular basis to make sure different devices are able to interact with each other whereas the existing hardware architecture can be used for different networks.

IoT middleware exists between the hardware and the applications which helps bring all the devices together to form a coherent unit and be able to exchange large amounts of data. These devices can perform new tasks without requiring to change or write new code for each service.

An IoT browser is necessary which can identify the mobile and dynamically changing, large amounts of data and suggest smart objects and interaction between these objects and a search engine exclusively for IoT that can search for objects which generates lots of information that is not constant. All these require the identification of proper internet protocols and be able to define new protocols which can answer different service scenarios.

Lack of security and privacy is also a challenging aspect in IoT. To make sure the network is secure, a number of encryptions exist for use. But most of them may not be appropriate for the devices or sensors because the encryption techniques may require high energy and high computational power to work whereas the sensors cannot allocate most of their energy towards security. Therefore, there is a need to make encryption algorithms faster. Advances are made in lightweight cryptography keeping IoT in mind. New encryption techniques like lightweight block ciphers, hash functions, stream ciphers are developed for low resource devices. Risks for IoT networks include attacks like Denial of Service (DoS) and Distributed Denial of Service (DDoS), eavesdropping and man-in-the-middle. [12]

2.5 Applications

As the uses of having IoT and the development of IoT technology and infrastructure are ever-increasing, the applications of IoT are increasing too. Some of the applications are in use as of today while the statistics show that IoT will be adopted into different areas [1] and hence affect most of our everyday lives. Some of the applications are discussed below.

2.5.1 Aviation

In the aviation industry, there exists a problem called the Suspected Unapproved Parts (SUP) which do not meet the requirements of the parts of the aircrafts [13]. This is a serious issue as there are accidents caused to airplanes due to SUPs. As IoT can be useful for identifying fake or duplicate products or parts, with the help of electronic tags, the products can be detected if they are real or duplicate and be able to go through their whole life cycle to see if there was any damage to it beforehand and if it was fixed or not by storing the details of all the parts in a secure database.

2.5.2 Automotive

IoT has migrated to the automotive industry with sensors and actuators attached to cars, trains, bicycles and buses. Before having vehicles was a norm, the technology used in cars was almost non-existent. Electric starters were only installed in cars in 1911 and then in 1925, cigarette lighters were added. Five years later, radio was also a feature in cars. Power steering was only introduced in 1956 whereas the cassette deck was installed in 1970 and finally the airbags in 1984. The more advanced features like CD players, computer dashboard and GPS navigation were introduced in 1980s and 1990s before the introduction of Bluetooth and USB ports. In today's world though, different parameters in these vehicles like tyre pressure, fuel consumption, speed limits, proximity to other vehicles and others are calculated and reported with the help of advanced sensors. Even while manufacturing these smart vehicles, RFID technology is used to streamline vehicle production and help improve quality control.

Andrew Meola from Business Insider [14] reports that by 2021, 94 million cars are expected to be manufactured and 76 million of those cars will be IoT featured and by 2020, 381 million cars will be IoT enabled and on the road. This increase in smart vehicles is not limited to cars. Public transportation will start to have smart buses and trains, if they are already not used. WiFi in vehicles, security cameras and electronic chargers are some additional features used.

Electric car manufacturers "Tesla" have planned to use solar roofs on their cars which have battery storage systems embedded into the vehicles. Self-driving cars are also something that are being tested now-a-days and there are claims of them being safer than manual driving. These self-driving cars are not only able to maneuver in traffic jams, but are also capable to drive themselves out of a parking spot and meet the driver of the car elsewhere with the use of a smartphone. If self-driving cars prove to be valuable, this idea can be applied into bigger vehicles like trains and even maybe aircrafts.

2.5.3 Telecommunications

In telecommunications with the help of IoT, different communication technologies like Near Field Communication (NFC), Global System for Mobile Communications (GSM), Bluetooth and WLAN can be merged with the SIM card.

With the help of advanced telecommunication technologies like 4G and 5G, IoT can be used in cross device integration, blockchain encryption, cloud communication and other mobile controlled applications. Blockchain technology is vital to protect one's data. It also plays a major role in how companies interact with each other, how they do business together and handle many transactions. This is really important, especially in Europe with the General Data Protection Regulations (GDPR) coming into force. [15]

2.5.4 Independent Living

To help aging people take care of themselves, IoT can help said individuals with the help of sensors and actuators attached to them. These sensors can monitor patient's activities and make decisions regarding their health with the help of machine learning and pattern recognition algorithms.

2.5.5 Pharmaceutical industry

Pharmaceutical industry has its own challenges. Counterfeiting products has many numbers of disadvantages in this sector. But this can be mitigated by placing smart labels to the genuine products and tracking them throughout their transport. It can also keep the supply-chain free from the criminals. These smart labels can also be valuable when tracking the storage conditions while being transported and also conveying the same to the consumers. Other information like expiration dates and dosages can be tracked. Physicians can also monitor the dosages and take note if the drugs are working well. If the results are not positive, they can change the dosages or the drugs itself.

Even during the manufacture of drugs, they have to be prepared in specific conditions like temperature, pressure and humidity. Different sensors can be placed during manufacture to oversee if the right conditions are met and to take note if the equipment for production is overused or underused.

2.5.6 Retail, Logistics and Supply Chain Management

Many challenges arise when dealing with logistics and supply chain management. Delays in transportation, very poor cargo conditions, errors in operations, thefts and other IT failures are too common and can decrease the profits and while increasing the costs to rectify the mentioned issues. A report by Forbes and Intel [16] claims that around 30% of the perishables never reach the grocers from the farms. With the help of connected logistics, IoT can not only avoid huge losses but also prevent wastage of food and beverages. Just like with the pharmaceutical delivery, temperature and humidity sensors can be attached to the shipment of the perishables or the crops and the owners or the drivers of the shipment can take note whether the parameters are above or within the threshold level and make necessary changes.

Issues can also stem from the lack of proper modes of transportation. To cut down losses due to such issues, Union Pacific which is the largest railroad company in the United States, have installed a variety of acoustic and visual sensors on the railway tracks which can detect minor changes and be able to predict train derailment. The railcars which are found to not be within the safety requirements are inspected and sent for servicing.

In retail stores with the use of RFID and smart shelves, the quantity of the stock can be optimized. Goods can be checked automatically and alert the retailer if the quantity present is less than required for that period of time. The threat of shoplifting can also be neutralized with the technology. Environmental issues like carbon footprint of the logistics services can also be tracked and necessary changes can be made to the equipment they use.

2.5.7 Manufacturing

In the manufacturing industry, unique identifiers can be used along with smart devices which can communicate with the whole manufacturing and production unit. By doing this, the whole life cycle of objects right from the production to disposal can be tracked and other manufacturing processes can be optimized. There is a lot of data about the specifications of products, the raw materials used, the conditions in which products are manufactured and the state of the equipment used but this data is not being taken advantage of. To increase efficiency, manufacturers are combining IoT and data analytics to increase the quality of the products built, the operational processes, maintenance of equipment and be able to predict any downfalls that may arise.

The current industrial era is termed as Industrial 4.0 which signifies that the manufacturing is “smart” and digital with the use of IoT, which is also known as Industrial Internet of Things (IIoT). Along with IoT, the use of big data, robotics and 3D printing is helping the digitalization of manufacturing. An IBM report [36] claims that the best-in-class firms (the firms that are the best in terms of performance which can be used as a benchmark for others) are implementing IoT technology to their units twice as fast as other industries. With this technology they are able to view the quality of data, the performance of operations, continuous status of operations and predict any issues.

These improvements will help in better quality of products manufactured. Best-in-class organizations once they implemented smart manufacturing, have observed a 7% increase in quality. They are not only good at better productivity, but also are very flexible in adopting new technology and standards to ease their manufacturing process.

2.5.8 Process Industry

In the oil and gas industry, IoT can be used to wirelessly scan different personnel in very complex petroleum operations such as tracking of containers, drilling pipes and monitoring of equipment. Most of the accidents in this industry occur due to lack of proper storage, separation of chemicals and different processes in it. With the help of wireless sensor nodes, the containers or the barrels can be tracked and monitored to avoid accidents. During drilling, the temperature of the oil can be really high and the pipes used for drilling may not handle the heat. Wireless sensors for temperature and pressure can be placed in these pipes to note and give out warnings if necessary.

2.5.9 Environment

In today's world where climate change, increase in pollution levels and other environmental issues are on the rise, IoT technologies can be used in green applications to conserve environment. Like in other applications, a network of sensors can be used to alert city officials or industries and factories in the event of rising emission and pollution levels.

Methane - which is the second leading contributor of global warming after carbon dioxide, is one of the leading components in industrial emissions. Southwest Energy - a natural gas producer in the United States along with IBM [17] is creating a methane monitoring system which can provide warnings if there is any leakage of the deadly gas using silicon photonics that can transfer data at the speed of light. With this fast response system and the wide array of sensors, they are hoping to trace pollutants and avoid any environmental damage as quick as possible.

A lot of companies in the oil and gas industry lose a lot of natural gas a year from leaks at disparate sites which require constant supervision to detect and rectify the leaks. Another company called Statoil from Texas have installed solar-powered methane detection sensors which work continuously and are self-powered, very cheap, require low maintenance costs and can also work for no less than five years. These gas leaks can be reduced by at least 40% [17] at a very cheap rate of one penny per thousand cubic feet of natural gas produced with the help of these solar-powered laser-based sensors.

Sensors can also be placed in the cities to locate pollution "hot spots" and can help educate people of the city. Energy can also be controlled at public spaces, offices, parks and even homes. Motion sensors can be placed at these locations and energy consuming devices like heating and cooling systems, lights, battery storage devices and chargers can be switched off when not in use. During the time of natural disasters, sensors can detect and warn authorities beforehand which will help people prepare avoid huge losses due to disasters.

2.5.10 Other applications

There are other applications where IoT can be used. In the transportation sector, IoT technology can be utilized for screening of passengers' luggage before boarding flights or trains and these bags if they go through international cargo, the security system can alert if the goods in the bags don't align with the security policies of different countries and thereby make sure that no security laws are broken. IoT technology can also be used by placing trackers in luggage or by using RFID. This becomes invaluable when a lost bag has to be tracked by airline operators. With the use of mobile phones and GPS, a person's location can be determined and the traffic situation in that area along with the use of Intelligent Transport Systems (ITS) can be monitored and this can be sent to people who plan to travel through that route.

The same idea can be implemented in agricultural industry as well. When one or more animals of a herd is missing, the tracker comes in handy to locate and detect lost animal in real time. Identifying animals is also important for the governments to note the rate of animal products manufactured such as meat and dairy products and regulate any animal diseases. This is easier with the use of IoT to get a real number.

IoT also has its uses in social settings. A lot of people have a user profile in one or more of the popular social networks like Facebook, Twitter and Instagram and share a few details about themselves in them. With IoT, a person's likes and dislikes can be extracted from their user profiles and scan for people with the similar interests who are close to them and connect each other or even inform the person about his/her friends who are close by.

2.6 Internet of Things (IoT) for Healthcare

While most of the applications for IoT were covered, there was one major application that was omitted from section 2.5 which is healthcare. IoT has numerous applications in healthcare and while this is benefitting, there are also many ways where it can go wrong. Since this work is centered around IoT for healthcare, it has a dedicated section.

2.6.1 Applications

In the last decade or so, internet connected devices were introduced in the healthcare domain. There was data flow from fetal monitors, electrocardiograms, blood glucose level sensors and temperature monitors. There was also use of smart-beds which can detect if the bed is occupied or if patient is trying to get up from it and provide required support and pressure to the patient that has occupied it. [19]

If IoT is used wisely, the healthcare domain can be improved a lot by delivering care for patients in different scenarios like acute patients who are treated in hospitals, people with long term illnesses who are treated in nursing homes and in community-based settings who are looked after at homes. With IoT, people in need of healthcare can be tracked along with equipment, medical supplies and different specimens. Vital parameters like heart-rates, doses of antibiotics and insulin amount can be monitored regularly with different sensors attached to patients and necessary action can be taken if anything is out of the ordinary. Laplante et al. [20] describe some of the following applications for a few health-related issues.

For a patient with an eating disorder or bulimia, sensors on the patient or in the patient's room can detect blood pressure, body temperature and even the odor of vomit. These details along with how hard the person is exercising, if beyond the normal rate, can help in managing and diagnosing the illness. Attaching a geolocation sensor on a person with Alzheimer's disease can help doctors, nurses, friends or close relatives prevent random wandering of the person and locate the person easily if that person is lost. With the help of other sensors, that person's blood pressure, level of diabetes can be monitored too.

In hospitals, the spread of diseases and infections is a major concern. Hygiene monitoring devices which are IoT enabled can help patients and other hospital staff prevent transmission. For pharmacy stores, IoT can help pharmacists keep track of the stock. Some pharmaceutical products need certain environmental conditions like temperature control and humidity where IoT can be useful. [21]

With IoT connected devices, health insurers can also keep track of conditions of patients for underwriting and claims operations. This will help them minimize claims which are frauds. They can charge less premiums for people who use IoT monitored devices and share the data with them. Precautionary health measures taken up by people can be rewarded by the insurers to maintain a good working relationship between the two parties.

Other wearables that people can use are devices which can improve a person's hearing. Hearables can transform the way people with hearing loss interact with the outside world. With the help of filters, features can be added to real sounds to differentiate between different sounds. These hearables can be synced with smartphones with the help of Bluetooth connection. [22] Some other body sensors are of the ingestible types where upon ingesting these pill-sized devices, they can keep track of the medication in our body and suppress symptoms and warn well ahead if they sense any disease-causing virus. Proteus [23] have come up with one of these sensors which can warn diabetic patients in advance.

Thync [24] have come up with mood enhancing therapy using bioelectronic sensors which are worn around the head and they send low intensity currents to the brain which improves the mood of a person. They are also currently performing clinical studies to get an approval from the FDA and establish the same kind of bioelectronic therapy in Psoriasis - a skin disease.

2.7 Challenges of IoT

Even though the technology has many uses and the rate of adopting IoT is high, IoT is still in its nascent stages. Because of this, there are numerous challenges that need to be addressed before it becomes usable in its entirety.

2.7.1 Security issues

With the amount of information devices need to become "smart" and for the IoT technology to work well, there is a primary need to secure the information by securing the devices. Since IoT is applicable to all kinds of devices that can be connected to other devices over a network, the specifications of these devices are not uniform. These devices range from various processing powers, memories, energy consumption rates and latency. Hence every device needs a specific encryption scheme to protect its data and at the same time have enough processing power to be able to perform its usual tasks without much bother. The task for designers and security professionals to choose an optimal encryption scheme for each device is a difficult one.

Applying updates to devices is necessary as the updates are not just for the firmware or the software but also for the security patches. But for some of the devices, the updates are not automatically performed. Therefore, the user has to take note of what all devices have some updates available. There might be a few devices which may not have any updates available which means the device is too old and the manufacturer stopped the updates. In these cases, it is essential for the users to upgrade the devices instead.

We have seen how many different uses the unique identifiers have in IoT. Hence identity management is key in IoT to protect the devices' integrity and to completely rid identity thefts and other illegal activities. IoT technology also uses a network to communicate between devices and this network has to be secure and safe. Web, mobile and cloud applications must be secured as they contain most of the data and are used to manage, access and process IoT devices. [18] Developers need to constantly keep in check if they have avoided the most common vulnerabilities (for example, OWASP Top 10) for each device they have built.

Authentication and authorization are vital for devices to avoid unauthorized users from accessing the devices. Devices should also have proper identity before accessing different networks, gateways and cloud applications. Usage of default or simple passwords can make the passwords easy to crack. In such cases, enabling two factor authentication can be helpful for proper authentication. Since this is an important issue, this has been the focal point of the thesis.

Lack of proper security and privacy protocols are the two main threats of IoT. Privacy is about individuals making sure that they have the right to control the information that is collected, maintained and used in various scenarios while security is about preventing access to one's information from users who are unauthorized and preventing removal or deletion of said information without the individual's consent.

According to Al-mawee [25], the main features of privacy are intractability (to not be able to lead every action performed by an individual back to him/her), unlinkability (to not be able to connect two or more actions), unobservability (to not be able to identify that an action has been performed, as opposed to protecting the user of the performed action), anonymity (protecting the information about the user) and pseudonymity (using pseudonyms like nicknames) and the main features of security are confidentiality (protecting information from unauthorized users), integrity (the data should not be tampered with), availability (the systems are always functioning and never suspended), access control (providing appropriate access to users with permissions and not denying them), authentication (ensuring the person is the one who he/she claims to be), non-repudiation (to not be able to deny the actions that the users performed) and notarization (examination of data by a third party which is trusted, that can assure the accuracy of the data).

2.7.2 Attacks

Attacks on healthcare IoT can happen a number of ways. The following are some of the attacks that can take place as per Ragupathy et al. [26]

- Eavesdropping is a form of passive attack (an attack that can learn from the information of the system but doesn't alter any system's parameters or resources) where the network information is extracted and can be read by the attackers and hence it is difficult to detect the attack.
- Radio Jamming is where radio waves are sent by an attacker at the same frequency as the other authorized sensor nodes.
- Message Injection is an active attack (an attack that directly affects the system's resources and performance) where the attacker sends messages to disrupt the records and probably overload the network.
- Node Destruction is where one of the sensor nodes is targeted to either entirely stop it from working or disrupt the network by affecting those sensor nodes which are in the same network as the targeted one. Sensor nodes can also be attacked to reprogram them.
- Denial of Service (DoS) is an attack where large amounts of data are sent to sensor nodes or even the data storage systems to make the targeted system out of order by making them consume a lot of energy in a short period of time.

- In Hello Flooding, the attacker uses a device with high transmission power to compromise every node and other sensors in its neighborhood.
- A malicious code which alters routing tables is inserted into the network in a Black Hole attack or a Packet Drop attack. It's a type of DoS attack where the router doesn't forward any packets but instead discards them.
- In the Gray Hole attack, which is a form of black hole attack, a node is infected with a malicious code and this node agrees to take part in route forming for exchange of data, but it actually does not.
- Another variant of the above attacks is the Wormhole attack wherein two sensor nodes are affected with malicious code which are connected and changes the route formation through these two nodes where the affected nodes collect the information of all other nodes.
- In Sinkhole Attacks, the malicious node tries to gather all the information from close to the base station by directing all the other nodes to send their information to the base station.
- Routing tables are modified by a malicious node which acts as multiple nodes in Sybil Attack.
- Message Alterations where a node alters the contents of the message and Slowdown where most or all of the nodes are kept awake in order to consume a lot of energy and slow down the network in the process.

Attacks on healthcare IoT can take place on a device, on the communication channel between the device and the data storage server or on the server itself. Of the above-mentioned attacks, eavesdropping, DoS are examples of attacks on the network or the communication channel. Node Destruction, slowdown are the examples of an attack on the device. Various data breaches on healthcare records are examples of attacks on the server.

While implementing perfect IoT security is a challenge (although any system being 100% secure is impossible), it is really important to have very good security measures for the healthcare sector as the stakes are higher and the repercussions far expensive. Lack of proper security measures can affect people's safety, and in some cases could also have life and death implications (someone dying from an attack is not recorded yet but pacemakers, infusion pumps have been hacked), not just financial losses.

The probability of an individual's health records being exposed is greater as they contain some valuable data. According to a Blackberry's report [27], about 80% of cyber-attacks are about stealing privacy-related information like personally identifiable information, personal health information and bank card or online bank information most of which are held in hospitals and 89% of healthcare workers claim that there was a breach because of adopting vulnerable IoT. Bill Siwicki's report [28] claims that about 71% of the ransomware attacks are caused because of very poor user practices. Although infusion pumps are the most widely used medical devices which are IoT connected, only 2% of the attacks are targeted on them. 45% of the attacks occur on imaging systems and 32% of the attacks occur on patient monitors. The individuals are not the only victims here. The organizations on the receiving end of the breach also lose reputation and suffer from brand damage which can lead to losses in the future.

Zingbox - a security company found that a bed in a hospital may have somewhere between 10 and 15 connected devices where a large hospital might have around 5000 beds. Each of those devices is a potential target for cyber-criminals and if there is a vulnerability even at one point, the whole hospital could be at risk.

In 2017, the United States Food and Drug Administration (FDA) confirmed that the transmitters of cardiac devices of St. Jude Medical, a global medical device company, which are implantable have vulnerabilities which when exploited have serious consequences. Once a device is hacked, the battery could be depleted and could also operate them to give incorrect pacing or shocks. [29]

In 2016, it was discovered that baby heart monitors which were developed by Owlet were not secure which can take lives of infants. Cesare Garlati, Chief Security Strategist of the PRPL foundation said that the connectivity element makes them weak and these kinds of devices need to be secure at the hardware layer [30]. In that same year, Johnson & Johnson, a US pharmaceutical company, whose insulin pumps were defective, warned its patients that they could be hacked, and the hacker can overdose the insulin amounts [31].

Shodan is a tool which can be used to scan and find more than 36,000 medical devices. A team of researchers found out a few security vulnerabilities in communication protocols of implantable cardiac defibrillators.

Without the usage of proper data protocols and standards, the devices are prone to attacks from cyber-criminals. There is a big discrepancy in data ownership regulations due to which patients' and doctors' personal health information can be compromised easily. With duplicate IDs, these criminals can buy drugs and medical supplies which cannot be obtained easily and sell them later for profits and can also file a fraudulent insurance claim in the patient's name.

2.7.3 Integration of multiple devices

Integration of a large number of devices in a network can also be cumbersome. Since there is no uniform standard when heterogeneous devices are connected to each other, there can be difficulties when a communication protocol has to be used. If the right one is not used, the network may not function at its maximum capacity by slowing down and can reduce necessary scalability that the healthcare sector needs.

2.7.4 Dealing with huge data

With the use of a lot of devices and communication protocols, data aggregation can be very difficult. Analyzing tons of data to develop machine learning and artificial intelligence techniques to gain some necessary and valuable insights of the data can be really difficult. This could get easier in a few years as processors could get more powerful then.

2.7.5 Cost

Even though IoT for healthcare is supposed to make healthcare cheap and affordable to people, it might take a few more years for that. With devices which need connectivity replacing the previous generation devices, hospitals and governments should invest in IoT to make healthcare cheap.

2.7.6 Legal issues

Legal issues may also arise from the use of IoT. Since IoT is not monitored by a single organization or entity, a shared governance approach should be undertaken which includes all the important stakeholders [11]. If there is an entity which claims accountability, then the governance can be improved if different sanctions are imposed.

Chapter 3

SystemC

As mentioned in Chapter 1, SystemC is a C++ library which is used to create a model of software algorithms, hardware architecture, System on Chip (SoC) and system level designs with accurate cycles. It is a system design language that answers questions about overall productivity for system designers. The hardware and the software components in SystemC work in tandem and resemble real-life conditions that can be imposed. This interplay between the software and the hardware components can give designers a thorough understanding of interactions and minute to large complexities of the entire system and helps them make necessary changes before the actual physical design is implemented to avoid huge losses and enable gains in productivity.

SystemC when used with SystemC Verification Library, most of the flaws and shortcomings in system design in other hardware programming languages are answered. Although there are tools and constructs in SystemC to enable register-transfer-level (RTL) modeling and synthesis, one major thing that stands out in its favor is the higher levels of abstraction.

One of the reasons why SystemC is chosen over other hardware programming languages is understanding design complexity and being able to implement it easily. With modern electronic systems gaining complexity where numerous sub-systems and components are present, the communication between them has become complex and it is important to get every interaction right in the design first before implementing it physically.

To answer questions about complex system design, SystemC aids by raising the level of abstraction of the design to build and maintain modern systems by supporting object-oriented design, modular programming, exception handling and data hiding [8]. Design reuse or external reuse of higher levels of abstraction increases productivity in RTL modeling and software design.

SystemC also supports team discipline which are the tools and techniques the engineers possess to gain productivity and interplay between engineers. These team discipline techniques can be used with reciprocity between software, hardware and the architecture teams throughout the design with a variety of tools. Project reuse (not to be confused with design reuse) is another advantage with SystemC where SystemC code that is written for a specific project can be reused in other parts of the same project or a new project. Abstraction levels can be changed by adding or removing details about design.

One more way to tackle design complexity is to automate design process. The Electronic Design Automation (EDA) industry provides required tools to help automate design processes which will help in writing numerous lines of RTL code required in modern systems.

Transaction Level Modeling (TLM)

Another big advantage of using SystemC is the use of transaction-based modeling approach. Communication and computation are separated in this method. SystemC supports modular programming where different components can be modeled as modules. Different processes can take place concurrently in SystemC and these modules can interact with each other in the form of transactions through connected channels.

The modules interact with each other through the use of TLM interfaces which are present in the channels as communication protocols. The processes to interact, have to access these interfaces through different ports in their modules. Transaction is just exchange of data or two modules being synchronized at any given instant which is decided by specifications of hardware and/or software used. [32]

The basic principle of TLM is to add enough specifications to the model that is required to develop the system components and sub-system for a defined task. Only once the necessary specifications are set, the designers and engineers can experience big gains in modeling speeds. And any changes that have to take place, can be made easily because the level of details added are not low.

With TLM, tasks that have some hardware implementations can be run early in the system development process. Even though TLM is a concept which is independent of the language used, with SystemC's features that support independent refinement of functionality and communication, it can be developed even further.

3.1 SystemC and C/C++

While C and C++ remain one of the most-used programming languages because it is easy to design software algorithms and interface specifications while they provide the data abstractions necessary to develop systems efficiently, required modules to model system architecture are missing. SystemC library of classes helps add these functions along with concurrency, hardware timing and reactive behavior. Since SystemC classes are of object-oriented nature, they cannot be added to the normal C language as the extensions to the language cannot be accepted. Hence C++, where the language can be extended through classes without changes to the syntax is chosen.

SystemC can be learned by understanding what the external constructs included do. It is easy to learn provided one understands C++ syntax and how C++ object-oriented concepts work. C++ tools which can enhance design productivity are available on the internet for a cheap price or sometimes for free which is another added advantage of using C++. Other factors like design reuse and imposing team discipline which were explained in the earlier section, are also a part of C++ and these techniques are used by developers.

Since SystemC is a C++ library of classes, a system needs to have a C++ and a SystemC environment that it can support. This environment includes, a platform that SystemC supports, a C++ compiler that is supported by SystemC, a SystemC library and a command sequence to compile and run (a make file can also be used). For this simulator, the platform used is Linux (Ubuntu 18.04.2 LTS specifically) and g++ which is a GNU C++ compiler.

The GNU C++ compiler in SystemC works in the traditional way. The compiler reads each file set where a file set usually contains at least one header (.h) file and one implementation (.cpp) file and creates an object for each file set. After different objects are created for their corresponding file sets using SystemC library, they are all linked to form an executable file which contains SystemC simulation kernel and the design functionality.

All the constructs that are available in SystemC which are required to design hardware are not usually present in a software programming language like C++ but they are implemented in the context of C++ language. Major hardware characteristics that are present in SystemC are different hardware data types, hierarchy in different modules that can manage structure and connectivity, a concurrency model, management of communication between concurrent executing units and a time model.

3.2 Installation of SystemC in Linux

Accellera Systems Initiative, which is a non-profit organization to create, support and advance system-level design, modeling and verification standards [33] in the electronics industry has the latest versions of SystemC which can be downloaded for free. A tar.gz file of "Core SystemC Language and Examples" of the version SystemC 2.3.3 which includes TLM was downloaded from the same source.

Then rest of the instructions are noted from systemc's github account. [34]

- Extract the contents of the downloaded file into a folder.
- Open Terminal in Linux. Move into the directory using `cd` command and extract the contents in the new directory.
- Create a temporary directory using `mkdir objdir` where objdir is the name of the temporary directory.
- Move into the temporary directory using `cd objdir`.
- Then `../configure`. This 'configure' command, while running detects the platform that the user is running and simultaneously prints out messages to inform different features that it is checking.
- To compile the package, use `make`, to optimize SystemC library.

- Then to install the package, use the command `make install`.
- And finally, to verify if the installation went as planned, `make check` command will compile and run different examples that are present in the “Examples” directory of SystemC.

3.3 Data Types in SystemC

Now that SystemC is installed and running, before compiling and running SystemC programs are looked at, it is important to understand what kind of data types are necessary in SystemC, excluding traditional C++ data types that are useful in defining variables.

The usual data types like ‘int’, ‘float’, ‘double’, ‘std::string’ and ‘bool’ are used in C++ for representing integers, floating point numbers, double (with twice the precision of numbers than float), strings and boolean values respectively and the keywords ‘unsigned’, ‘unsigned long’, ‘short’ are used to describe unsigned numbers only, unsigned long numbers (32 bit numbers as opposed to 16 bit numbers) and short numbers (16 bit numbers) respectively.

3.3.1 Integers

In SystemC, **sc_int** and **sc_uint** are used to define data types of integers and unsigned integers with the bit size in angular brackets with the name of the variable. The bit sizes or bit widths for these SystemC data types should be between 1 and 64.

For example, **sc_int<16> name;** is used to define a variable called ‘name’ with bit size of 16 and **sc_uint<32> person;** is used to define a variable called ‘person’ with bit size 32.

To define integers with higher bit widths than 64, SystemC has **sc_bigint** and **sc_biguint** data types. Although these data types provide higher bit sizes, the processing speed will have to be compromised. Hence it is advised to not use these data types for bit sizes of 64 or less.

For example, **sc_biguint<80> speed;** is used to define an unsigned integer variable named ‘speed’ of 80 bits and **sc_bigint<128> distance;** is used to define a signed integer called ‘distance’.

3.3.2 Boolean and Multi-Value

sc_bit is used to define ones and zeroes and for longer bit vectors, **sc_bv<bit_size>** is used where the size of the variable has to be defined in the angular brackets. Since these data types are exclusively for bits and not for integers or floating-point numbers, they cannot be used for arithmetic operations. But they can be used to perform standard bitwise operations like and, or, xor.

For example, **sc_bit flag;** is used to define a bit value called 'flag' and **sc_bv<7> display = "1010110";** is used to define a variable called 'display' whose value is 1010110.

In other hardware description languages like Verilog and VHDL, a bit of information can be represented in more than two states i.e. 1 and 0. There are two other states - 'z' and 'x'. 'Z' is used to signify high impedance or to imply that there is an unconnected physical wire and 'X' is used to describe a physical wire which has an unknown value. This unknown value could arise due to many reasons one of which could be that the wire has more than one driving source or the wire may not have been assigned previously. [35]

SystemC represents these multi-state values using **sc_logic** for a single bit value and **sc_lv<bit_size>** for bit vector types where the bit size is mentioned in the angular brackets. These data types are slower than **sc_bit** and **sc_bv** due to overhead.

For example, **sc_logic flag (SC_LOGIC_Z)** is used to define a variable called 'flag' and assign the value 'z' to it and **sc_lv<6> display ("XX00Z1")** is used to define a variable called 'display' and assign the value XX00Z1 to it.

3.3.3 Fixed-Point Data Types

To represent not just whole numbers but also fractional components, fixed-point data types are used. This will especially be useful when working with Digital Signal Processing (DSP) applications. DSP processors which are integer-based do not support floating point numbers hence fixed-point numbers are used. A few parameters must be set to use fixed-point numbers. These parameters are word length, integer word length, quantization mode, overflow and number of saturation bits. Out of these, word length and integer word length should be set by the user.

The syntaxes of these data types are **sc_fixed, sc_ufixed, sc_fix** and **sc_ufix**. **sc_ufix** is the same as **uint**. The variables with data types 'fixed' in them are pre-defined data types and will have parameters that are set during compile-time constants. Once they are set, they cannot be changed after compilation where the other data types with just 'fix' in them can be changed dynamically.

If the keyword **_fast** is added to the above-mentioned data types, then these variables work at a faster rate. But the precision of these variables is limited to 53 bits only.

Some of the data types of SystemC are faster compared to the others. Traditional C++ data types like **int, double** and **bool** are fastest. SystemC exclusive data types from fastest to slowest are:

sc_int<> and **sc_uint<>; sc_bit** and **sc_bv<>; sc_logic** and **sc_lv<>; sc_bigint<>** and **sc_biguint<>; sc_fixed_fast<>**, **sc_fix_fast**, **sc_ufixed_fast<>** and **sc_ufix_fast; sc_fixed<>**, **sc_fix**, **sc_ufixed<>** and **sc_ufix**.

3.4 Modules, Methods and Threads

3.4.1 Modules

All SystemC programs start with a main function - **sc_main()** just like how C++ programs start with main(). Most of the programs with the main function will be named main.cpp. In this main function, the code is executed in three stages. In the first stage which is elaboration, data structures are initialized and those structures which are necessary for the interconnections of the system are connected. In simulation, which is the second stage, code of the model is executed and in the final stage called post-processing which is an optional stage, the code will format reports and take care of the simulation results.

Components in SystemC have the same properties as classes in C++. If the model to be designed is a complex one, usually there is a few components that make up the model. These components could be of the hardware type, software type or of physical nature. These components are hierarchical. Bigger components may contain smaller components in them. To represent these components the macro **SC_MODULE** is used. The syntax of the module is:

```
#include<systemc.h>
SC_MODULE(name_of_the_module)
{
    MODULE CODE;
};
```

The “#include<systemc.h>” statement includes all the SystemC libraries into the program. In this module, ports, constructor functions, destructor functions, processes, channel instances, data instances are defined. Of these, the necessary function that has to be defined is the constructor function. Others are defined depending on the module function. The constructor function that is supposed to be defined in SC_MODULE is SC_CTOR (name_of_the_module). The name of the module in the constructor function should be the same as in the module definition. This constructor allocates and connects sub-designs, registers processes with SystemC kernel and provides sensitivity.

Now that we have seen how to define modules and constructor function in a module, here is an example of a basic SystemC program - printing “Hello World”.

```
#include<systemc.h>
SC_MODULE (hello_world)
{
    SC_CTOR (hello_world)
    {
    }
    void hello()
    {
        cout << "Hello World. \n";
    }
};
int sc_main (int argc, char* argv[])
{
    hello_world say ("");
    say.hello();
    return(0);
}
```

3.4.2 Methods and Threads

Once the SystemC simulator is run until it stops, the code is initiated in ‘processes’. A process is a function in SystemC’s SC_MODULE which is called by the scheduler of SystemC’s simulation kernel. A thread in SystemC is a form of a hardware process. These threads run concurrently in SystemC, an advantage over C where the threads run in sequence. For hardware processes, these threads can be made sensitive to signals, clock edges or for a particular amount of time. These processes or threads are always active, so they don’t have to be called by the user.

SystemC has three types of threads which are **SC_METHOD()**, **SC_THREAD()** and **SC_CTHREAD()**.

SC_METHOD()s are threads which are executed once in their entirety when every sensitive event is triggered. They are run continuously and they are equivalent to Verilog’s @ always block. They are supported for synthesis and are useful for combinational logic and sequential logic that can be performed once, for example in counters. One iteration of this thread takes exactly one clock cycle. For example, in a simple two-input AND operation, for every clock cycle, the inputs of the function change and accordingly the output changes too if the sensitive event is rising or falling edge of the clock.

SC_THREAD()s are threads which are executed once at the start of the simulation and suspend themselves when they are done and for this reason they are not used much. But for them to run continuously, they can have an infinite loop in them which can run for a fixed time interval. They are equivalent to Verilog’s @ initial block. They are not supported for synthesis. They are used for testbenches to describe clocks or startup sequences.

In SC_CTHREAD(), C stands for clock or clocked thread. They are run continuously in a program. They can be sensitive to a clock edge and they are synthesizable. They are different from SC_METHOD() in that they can take more than one clock cycle to execute which makes them useful for a high level design. Hence SC_CTHREAD() is used in most of the complex designs where there are large blocks of code with multiple operations and control.

3.4.2.1 Concurrency

It was mentioned earlier that SystemC processes are concurrent. But in reality, the execution of these processes is not concurrent. Although the concurrency is not pre-emptive, simulators of each process execute a snippet of code and then intentionally let go of control to let other pieces of code from other processes execute. Decisions like this are taken up by the simulator kernel. Its responsibilities mainly include starting processes and then deciding the order. Once this is done, it is the job of the processes themselves to suspend and allow other processes to run.

It takes three stages for the code to execute as discussed in section 3.4.1. The simulation kernel is called by **sc_start()**, which occurs after the elaboration phase. Then the simulation phase starts where the processes that are defined earlier are started and placed in a ready pool. A process is executed from the ready pool until the **wait()** statement (if there is one) and is suspended and then the next process executes. This method continues until all the processes are complete (until the **return** statement is executed).

3.5 Ports and Signals

There is a big need for communication between modules for complex hardware designs. The best way to communicate is to make sure that right processes are executed at any given order. And the easiest way to establish communication between modules is to define ports in each module and have a signal or a channel connect these ports.

3.5.1 Ports

To send and receive data from different processes in different modules, ports are used. A module may have different ports in it and each port has to have a defined mode - in, out or inout. If the mode is ‘in’, the port can only receive data and if the mode is ‘out’, the port can only send data to other ports. ‘Inout’ means the port can both send and receive data. The type of data that can be sent or received has to be defined too while defining ports. These data types can be C++ data types, SystemC data types or user defined type. Each port will also have an identifier or a variable with which it can be associated.

The Figure 2 is an example of a module with five ports in it. The ports ‘a’ and ‘b’ are output ports which are on the right, the ports ‘c’ and ‘d’ are input ports which are on the left and the port ‘clk’ is an input port which provides clock signals.

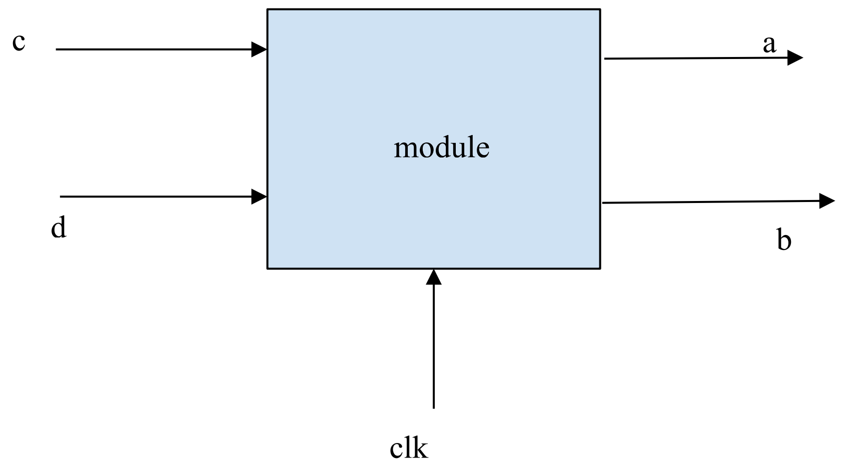

Figure 2. Example of a module with input and output ports

This particular module can be defined in the following way in SystemC:

```
SC_MODULE(module)
{
     sc_out<bool> a;
     sc_out<int> b;
     sc_in<char> c;
     sc_in<int> d;
     sc_in_clk clk;
};
```

These ports use functions to read and write data. To read from an input port, the function “.read()” is used whereas to write to an output port, “.write()” is used. In the above example to read data from port ‘d’,

x = d.read(); statement will read integer type data, since port ‘d’ is of type ‘int’ and store the read value to a variable ‘x’. And to write data to port ‘a’,

a.write(y); statement will write ‘y’ of type ‘bool’ to the output port ‘a’ since ‘a’ is of the type ‘bool’.

3.5.2 Signals

Signals are analogous to physical wires which connect two devices together. In SystemC, they can be used to connect two ports in order to exchange data between them. While signals provide a medium for data to be carried over to and from different ports, the connected ports determine the direction of data to be transferred.

The keyword used to define a signal in SystemC is **sc_signal<DT> variable;** where DT is the data type of the information that the signal carries and ‘variable’ is the identifying name of the signal. The data type of the signal should match the data types of the ports it is interconnecting.

Figure 3 is an example of two modules ‘A’ and ‘B’ which have various input and output ports and different signals to connect them. The definition of these signals can be done in the following way assuming all the data that is transferred are integers.

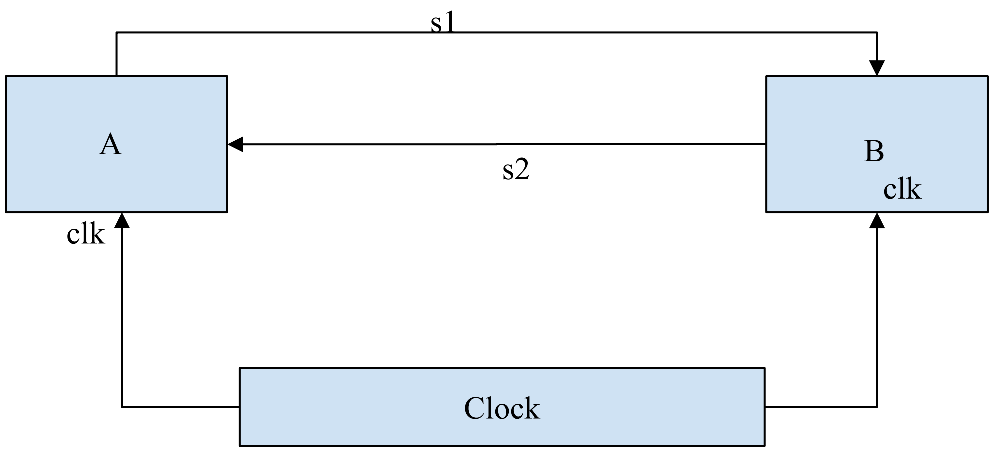

Figure 3. Example of Signals connected to ports of different modules

```
sc_signal<int> s1 (“Signal-1”);
sc_signal<int> s2 (“Signal-2”);
```

The common clock signal for all the three modules can be defined as:

sc_clock name("name", period, duty cycle, first edge); where sc_clock is built-in hierarchical channel for clocks. "name" is the name given to the clock and in the parenthesis, the first parameter is again the name of the clock, the second parameter is the time period in time units, then the duty cycle value and finally the time at which the first edge of clock will occur in time units.

3.6 An Example of a SystemC Program with Ports and Signals

The system in the figure 3 can be built in SystemC by defining each module first in its own header file and then using signals to connect them in the main file. Each module also has its own .cpp file to define the processes and functions that take place in them.

ModuleA.h

```
SC_MODULE(module_A)
{
     sc_in_clk clk;
     sc_in<int> a;
     sc_out<int> b;
     SC_CTOR(module_A)
     {
          SC_CTHREAD(connect1
, clk.pos());
     }
     void connect1();
};
```

ModuleB.h

```
SC_MODULE(module_B)
{
     sc_in_clk clk;
     sc_in<int> c;
     sc_out<int> d;
     SC_CTOR(module_B)
     {
          SC_CTHREAD(connect2
, clk.neg());
     }
     void connect2();
};
```

ModuleA.cpp

```
#include<systemc.h>
#include "ModuleA.h"
void module_A::connect1()
{
     int n;
     while (true)
     {
          n = a.read();
          n = n + 1;
          cout << "n = " << n
<< endl;
          b.write(n);
          wait();
     }
}
```

ModuleB.cpp

```
#include<systemc.h>
#include "ModuleB.h"
void module_B::connect2()
{
     int n;
     while (true)
     {
          n = c.read();
          n = n + 2;
          cout << "n = " << n
<< endl;
```

```
          d.write(n);
          wait();
     }
}
```

Main.cpp

```
#include <systemc.h>
#include "ModuleA.h"
#include "ModuleB.h"
int sc_main (int ac, char
*av[])
{
               sc_report_hand
     ler::set_actions("/IEEE_
     Std_1666/deprecated",
     SC_DO_NOTHING);
     sc_signal<int> s1
("Signal-1");
     sc_signal<int> s2
("Signal-2");
     sc_clock clock("clock",
2, 0.5, 0.0);
     module_A m1 ("M1");
     m1.clk(clock);
     m1.a(s1);
     m1.b(s2);
     module_B m2 ("M2");
     m2.clk(clock);
     m2.c(s2);
     m2.d(s1);
     s1.write(0);
     s2.write(0);
     sc_start(100, SC_NS);
     return 0;
}
```

To compile the set of programs, SystemC uses a C compiler since it is a C++ class library. In this simulator a GNU C compiler - g++ is used which is present on almost all the Linux machines. To check where g++ is present in the Linux system, use the command **which g++** which gives out the location. The compiler needs an executable name which can be provided with the **-o** switch and the list of .cpp files that need to be compiled. The path to design source and SystemC source should also be provided in the compilation command along with the **-I** switch. Finally, using the **-L** switch, SystemC and C++ archive files' paths have to be provided in the same command.

To make it easier for type in the locations of different directories, the path to where SystemC is installed can be saved to a variable. For example,

export SYSTEMC_HOME=/home/user/systemc-2.3.2/, will save the path to the SystemC directory to the variable **SYSTEMC_HOME**.

The compilation command for the above example is:

g++ -I. -I$SYSTEMC_HOME/include -L. -L$SYSTEMC_HOME/lib-linux64 -Wl, -rpath=$SYSTEMC_HOME/lib-linux64 -o connection ModuleA.cpp ModuleB.cpp Main.cpp -lsystemc -lm

The dots (.) after -I and -L means local. This tells g++ to look into the local directory for the source files. The -lsystemc and -lm are the archives that the SystemC compilation will need. The former is the SystemC archive and the latter is the math library archive. Once the above command is run, an executable called "connection" is created.

The command **./connection** will run the executable and provide output in the console.

Instead of typing the command for compiling and running the program every time, a change is made to it, a makefile can be created and saved in the same directory as the program is run. In this makefile, different variables are created and to these variables necessary values are stored which are needed for the compilation. This makefile can be run by simply typing 'make' in the command line which is equivalent to compiling the program. A makefile can be created in the terminal by typing

$vi makefile

In the makefile add the following lines,

```
SYSTEMC_ARCH=linux64
LIB_DIRS=$(SYSTEMC_HOME)/lib-$(SYSTEMC_ARCH)
INCLUDE_DIRS = -I. -I$(SYSTEMC_HOME)/include
HEADERS = ModuleA.h ModuleB.h
SOURCES = Main.cpp ModuleA.h ModuleB.h
DEPENDENCIES = \
                   Makefile \
                   $(HEADERS) \
                   $(SOURCES)
LIBS= -lsystemc -lm
TESTS= main
all: $(TESTS)
    ./$(TESTS)
$(TESTS): $(DEPENDENCIES)
                g++ $(INCLUDE_DIRS) -L$(LIB_DIRS) -Wl, -
rpath=$(LIB_DIRS) -o connection $(SOURCES) $(LIBS)
```

This makefile will not only compile the program but also run it and show the output.

Chapter 4

Simulator Design and Implementation

Moosavi et al. [6][7] proposed a scheme where the data transfer is secure between medical sensors and devices which monitor patients' vital signs and other medical parameters and the end-users who read them. This scheme, in order to be secure uses the concept of smart e-health gateways which can perform some of the tasks that sensors usually perform thereby easing some of the workload of the sensors. Proper end-user authentication and authorization architecture is employed by using certificate-based DTLS handshake between the end-user and the smart gateways. Before the advantages and improvements of this scheme are realized compared to other ways of secure data transfer, the target system to be replicated is described.

4.1 The Target System

To tackle the safety of the patients who are undergoing treatments, security of the systems so that there is continuous monitoring of the patients and dependability of the system so that the results obtained are accurate and prompt with a reliable service so that the requirements are met, the latency of the smart components has to be predictable and the communication with the computing layer has to be reliable. The conventional methods not being helpful with the requirements where the latency is high and communication with cloud not as reliable, different measures are needed which can improve the system.

To address the issues of cloud computing, an additional layer is introduced known as “Fog Layer”. This fog layer is close to both the sensor nodes and also end-users. Using smart gateways in fog layer can improve real-time interaction, mobility of patients where sensors attached to patients can communicate with the nearest gateways, interoperability, heterogeneity where the devices in fog layer range from end-user devices and access points to edge routers and switches. These services can be implemented in a variety of devices, like smart phones and edge routers.

This system uses three layers (see Figure 4) for the information to pass from sensors to end-users. The information gathered is from medical sensors and devices which are attached to the patient’s body. Also, extra information like time, date and location may be present along with the data.

The first or the bottom most layer is the device layer which consists of actual physical devices which are planted on a patient and medical data is collected. These devices sense, collect data, communicate periodically and capture bio-medical and context signals from the person and the room. With these signals, different problems can be diagnosed. These signals are then sent to the middle layer through wireless or wired protocols.

The fog layer, which is the middle layer consists of smart gateways which are interconnected. While cloud computing can be helpful with creating and maintaining data storage systems and private servers, the freedom and the productivity they offer to web applications and very minimal knowledge requirements of the hardware and infrastructure, the computational level and the data storage needed for them is high and it can be a disadvantage with low-latency applications, like emergency care. Hence fog layer is introduced which extends the cloud applications to the edge of the network.

The gateways used here are smart, e-health which can adjust to different communication standards and protocols. They receive data from different networks, convert one protocol to the other and provide different services. They can also store the data they receive temporarily. They also can adjust to different issues like energy efficiency, scalability and reliability.

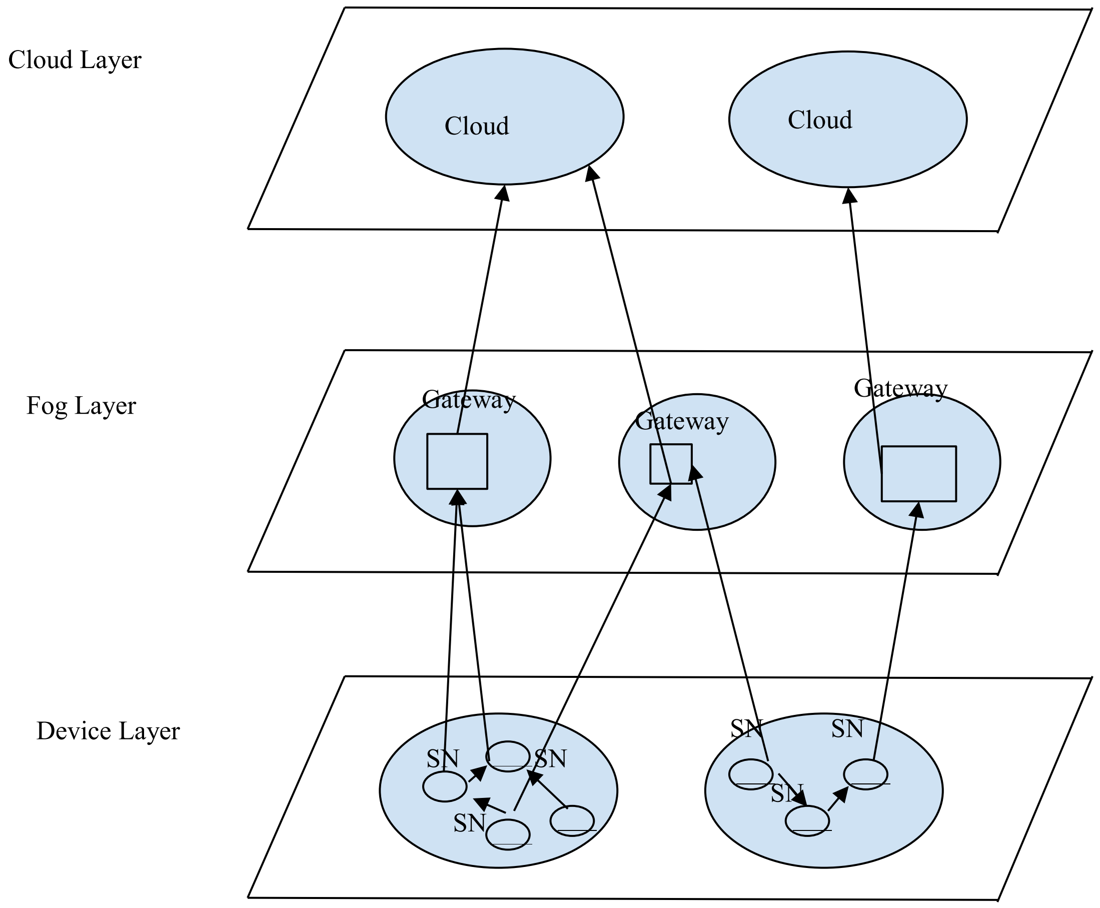

Figure 4. Three-layer architecture of Healthcare IoT

The top most layer is the cloud layer where the data is stored in a database and data synchronization is done with the remote healthcare database server. This layer also has tasks like broadcasting, data warehousing and big data analysis. The data that is received can be categorized into two types, public and private. Public data may include blood type and private data can include DNA.

4.2 Necessities of a Secure Communication System for Healthcare IoT

There is a few characteristics that a secure communication in healthcare IoT should possess. Some of those are:

- Data confidentiality between all the parties in the system. To prevent private and important information from falling into the wrong hands, strong encryption techniques are required to even make it difficult for cyber-criminals who eavesdrop and target devices.

- Data integrity should also be addressed. The data that is sent from the first device should match the data received by the end-user. To ensure data integrity, usually a Message Authentication Code (MAC) or a Cyclic Redundancy Check (CRC) are employed.
- Proper authentication and authorization should be done by both the parties involved in communication to identify each other. This is important to make sure that confidential information is not accessible by unauthorized users or users without proper authentication.
- Data freshness is important to prove that old data is not being transmitted again by an adversary.
- Availability of medical sensors and other services means that they can provide necessary functions whenever requested by an authorized user. This might prove to be difficult as DoS attacks can limit the performance and minimize the power.
- Scalability also needs to be taken care of as quickly as possible if there comes a requirement of expanding or downsizing the existing system so that the communication system is active for as long as possible. Since the sensors need as much energy resources as possible to be devoted to receiving and sending data, care should be taken that the cryptographic algorithms employed must be lightweight.
- Finally, end-to-end security is an important feature where sensors and end-users can securely communicate with each other.

4.3 Mutual authentication and authorization using certificate-based DTLS handshake

In this particular scheme, the communication is strictly not one-way. Not only do the medical sensors collect and transfer data to the end-users, but the end-users can also choose to control the sensors through Internet. In order to protect this data transfer by blocking malicious activities at the edge of the network and by employing mutual authentication and authorization techniques, Datagram Transport Layer Security (DTLS) protocol is applied between the end-user and the smart gateways in the fog layer. These smart e-health gateways take part in this authentication and authorization process on behalf of the sensors, thereby ensuring that the sensors don't take up too much of their energy in this process and the smart gateways and the end-users have enough resources to perform this heavyweight security protocol and validation of certificates.

Any IoT application might have one of four security modes. In "NoSec", the DTLS handshake is disabled without any protocol level security but IPsec can be implemented as network layer security. “Symmetric Key-based DTLS” has DTLS enabled using symmetric key algorithms. “Public Key-based DTLS” has DTLS enabled and the sensors used have an asymmetric key pair. Finally, in “Certificate-based DTLS”, DTLS is enabled where the sensors use an asymmetric key pair and the public key is included in the X.509 certificate which is signed by a Certificate Authority (CA). For this research, certificate-based DTLS is used.

Refer to Figure 5 for the certificate-based DTLS handshake between the smart e-health gateway and the end-user. This is the sequence of events that happen during the handshake.

- The handshake begins with a message from the client or the end-user which is **ClientHello**. In this message, the security parameters for the handshake are sent. The pre-master secret key is then computed using these security parameters. The server responds with a **HelloVerifyRequest** with its security parameters. The client then sends a **ClientHelloVerify** message to the server with an additional cookie.
- The next flight of messages originates from the server. The first message is **ServerHello** which has the cipher suite that was negotiated and a random value from the server which is used later to calculate the master secret key. This cipher suite that is proposed by the server has to be supported by the client or else the handshake fails with a **HandshakeFailure** message. Then, the server sends a message with its certificate chain which has the server’s public key. The **ServerKeyExchange** is sent with cipher suites that need more variables to compute the master secret key. The cipher suite used in the research is Elliptic Curve Diffie-Hellman (ECDH) and Elliptic Curve Digital Signature Algorithm (ECDSA). Finally, the flight of messages is concluded with a **ServerHelloDone** message.
- The client sends the next set of messages to the server. The client’s certificate is sent first with its public key, then **ClientKeyExchange** is sent with variables that are needed to compute the master secret key. **CertificateVerify** message is then sent which proves that the client possesses the private key which is a pair to the public key present in the certificate. **ChangeCipherSpec** message is sent by the end-user to tell the smart gateway that the following messages will be encrypted using the cipher suites with the secret keys which were decided earlier. Finally, the **Finished** message has the hash values of all the earlier messages to prove that both of them have been communicating unmodified messages and the handshake is successful.

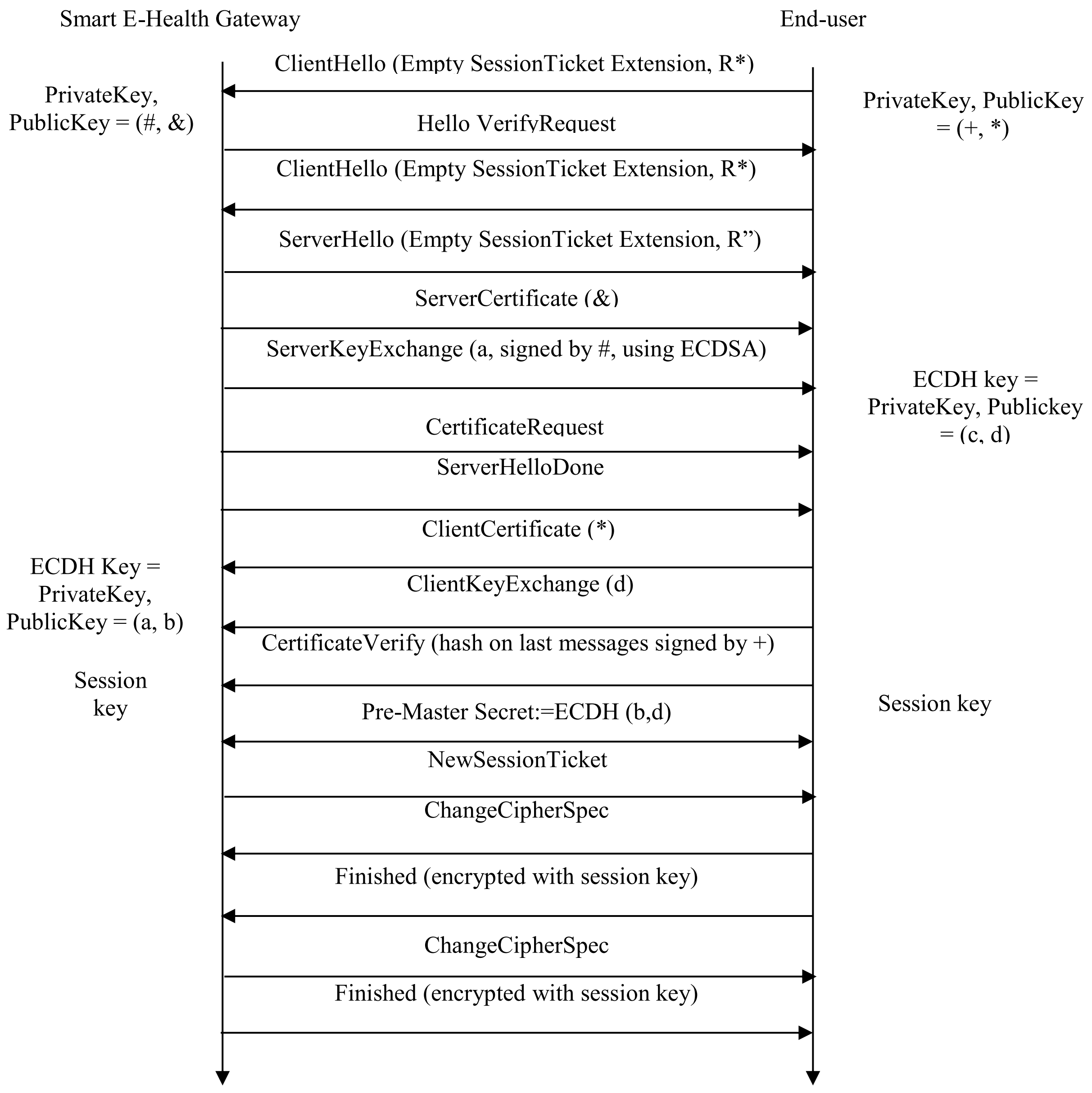
Smart E-Health Gateway
End-user
ClientHello (Empty SessionTicket Extension, R*)
PrivateKey, PublicKey = (#, &)
Hello VerifyRequest
PrivateKey, PublicKey = (+, *)
ClientHello (Empty SessionTicket Extension, R*)
ServerHello (Empty SessionTicket Extension, R")
ServerCertificate (&)
ServerKeyExchange (a, signed by #, using ECDSA)
ECDH key = PrivateKey, Publickey = (c, d)
CertificateRequest
ServerHelloDone
ClientCertificate (*)
ECDH Key = PrivateKey, PublicKey = (a, b)
ClientKeyExchange (d)
CertificateVerify (hash on last messages signed by +)
Session key
Session key
Pre-Master Secret:=ECDH (b,d)
NewSessionTicket
ChangeCipherSpec
Finished (encrypted with session key)
ChangeCipherSpec
Finished (encrypted with session key)

Figure 5. Certificate-based DTLS handshake between smart e-health gateway and the end-user[6]

- The final flight of messages from the smart gateway include the **ChangeCipherSpec** and **Finished** messages. Once this is done, the client and server use this connection to send and receive information securely.

4.4 Secure Communication Scheme

The certificate-based DTLS handshake that was detailed in the section 4.3 is to mutually authenticate and authorize the smart gateways and the end-users but the sensors and devices have not been authorized yet. In order to provide end-to-end secure communication between the sensors and the end-users, a session resumption technique is proposed by Moosavi et al. [7]. This technique is an extended form of the handshake and it doesn't include encrypted session states of the DTLS handshake towards end-user. By doing this, overhead on sensors is reduced because all the high energy consuming tasks are performed in the handshake.

In the research the type of DTLS session resumption technique used is DTLS session resumption without server-side state in which the server (sensors and other devices) need not use encrypted session state while communicating with the client (the end-users) after the DTLS handshake takes place. The session resumption is requested by the client in the handshake in the ClientHello message which has an empty session ticket extension whereas the server does the same in ServerHello message.

Once the handshake is complete, the smart gateway updates the sensors regarding the authenticity of the end-users and that the handshake is successful by encrypting it in AES-CCM algorithm. Then the actual session resumption takes place according to Figure 6.

- The end-user starts the sequence of messages with a **ClientHello** which has the session resumption extension and a random value R*. The sensor then employs the session update which is in an encrypted format from the smart gateway and resumes communication.
- The server (sensor) then decrypts the ticket and cross-checks it to see if it corresponds to the pre-master secret.
- Once verified, the sensor then sends a **ServerHello** message an empty session resumption ticket with a random value R". Along with that a **NewSessionTicket** is sent which has the current master secret which is calculated using a Pseudo Random Function (PRF). It also has a lifetime value which indicates the time in seconds until the session ticket expires.
- Then by **ChangeCipherSpec** messages from both the server and the client, the new key is then used to encrypt the communication channel.

- With the **Finished** messages, the validity of the keys and the integrity of all the messages are cross-checked.

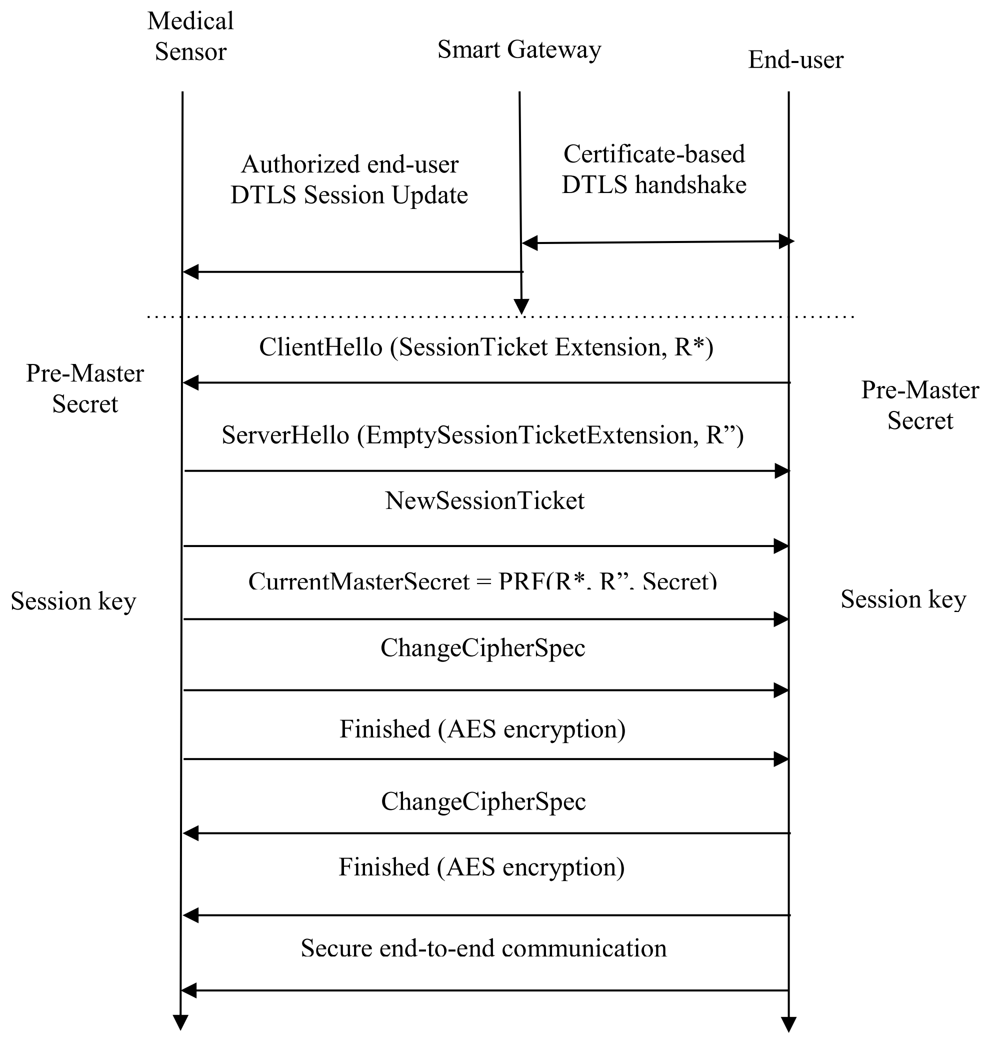
Medical Sensor
Smart Gateway
End-user
Authorized end-user DTLS Session Update
Certificate-based DTLS handshake
ClientHello (SessionTicket Extension, R*)
Pre-Master Secret
Pre-Master Secret
ServerHello (EmptySessionTicketExtension, R”)
NewSessionTicket
CurrentMasterSecret = PRF(R*, R”, Secret)
Session key
Session key
ChangeCipherSpec
Finished (AES encryption)
ChangeCipherSpec
Finished (AES encryption)
Secure end-to-end communication

Figure 6. Session resumption based end-to-end security[7]

4.5 Results of the research

To track the results of the certificate-based DTLS handshake and the DTLS session resumption technique, medical sensors, UT-GATE smart gateway which is made of Pandaboard, CC2538 module and SmartRf06 board made by Texas Instruments, remote server and end-users are used.

To implement DTLS, TinyDTLS was used which is an open-source DTLS symmetric key based version which can be extended to certificate-based DTLS also with session resumption.

When transmission overhead and latency of the implemented handshake method is compared to other widely used methods, there is an improvement. When the messages are exchanged in IEEE 802.15.4 radio links, the average transmission overhead in 100 different runs is 1190 bytes in 18 different packet fragments (packets are fragmented because the message size is too large) which is a 26% improvement over delegation-based architecture which needed 1609 bytes over 24 fragments [7] and the latency or the time taken by a packet to go from one point to the other, 5.001 seconds as opposed to 6.08 seconds from the delegation-based architecture.

Other improvements due to session resumption using DTLS without server include reduction of processing time from the client and the server and reduction in run-time performance and energy consumption compared to certificate-based DTLS. The processing time of client and server is 45 ms and 160 ms respectively whereas it is 3744 ms and 5690 ms respectively in certificate-based DTLS. The run-time performance and the energy consumption are 205 ms and 8.87 mJ respectively whereas it is 9434 ms and 315.79 mJ for the certificate-based DTLS handshake. The final improvements can be noticed in RAM and ROM overhead. For this research RAM and ROM overhead are 3.51 kB and 14.29 kB respectively and they are 7.8 kB and 41.1 kB respectively for certificate-based DTLS.

4.6 Simulator Architecture

To replicate the above-mentioned work as a SystemC simulator, an architecture needs to be designed as a base. Number of sensors, gateways and end-user were decided to simulate the research. Modules were created and defined in SystemC for each component. The initial design comprised of 5 medical sensors, 1 smart e-health gateway and 1 end-user. A SystemC class was defined for each of these components. The connections between the medical sensors, smart e-health gateways and the end user are shown in Figure 8.

The device layer consists of all the five medical sensors which get the input from a file in our simulator instead of a patient. These sensors are connected to another module which represents smart gateway, present in the fog layer which performs mutual authentication and authorization on behalf of all the sensors with the end-user. Also, the data collected by the sensors is sent to the smart gateway and gateway transfers the data to the end-user class which is theoretically present in a cloud server or a database and then into a file where the same data can be read.

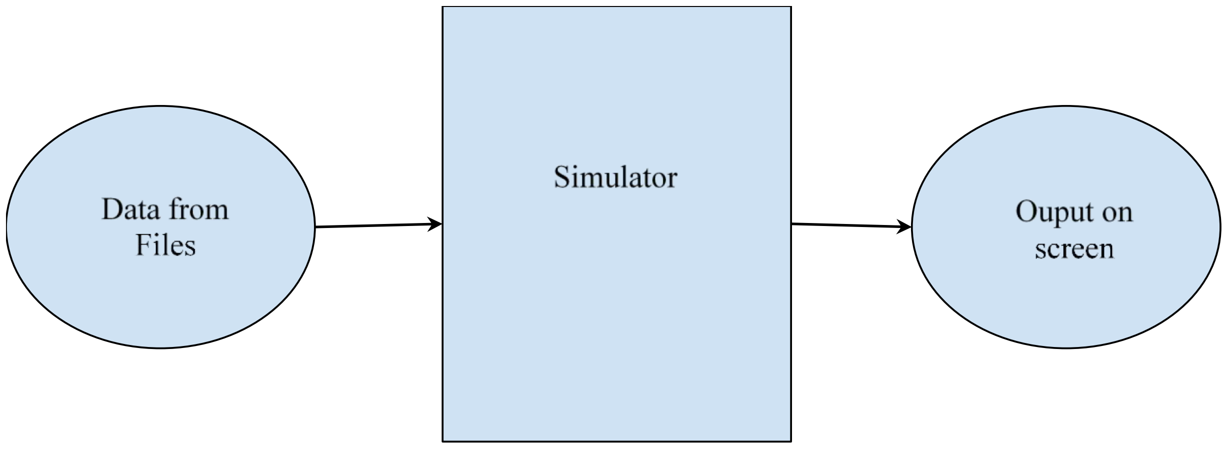

Figure 7. Showing data flow to and from the simulator

The connections between modules are done using the concept of ports and signals as discussed in Section 3.5. Input and output ports are defined for each module and signals are defined to make necessary connections between the sensors and the gateway and from gateway to the end-user.

The data to the sensors comes through files which are created, and random values are added to it. And once the values reach the end-user module, they are printed on the screen. Figure 7 shows how the values enter and leave the simulator. The direction of the flow of data is one-way in this case. Since the gateway has to receive communication from the sensors, perform certificate-based DTLS handshake with the end-user and transfer data to the end-user, it has a lot more ports and signals defined.

4.7 Simulator Implementation

Figure 9 shows the class diagram of the simulator. As mentioned in Section 4.6, each module has its own class and various functions are defined in each module. All the modules use the SC_MODULE() macro (see Section 3.4.1) to define the classes, the input and output ports used, the constructor function, the threads that are defined in the constructor functions and other member functions. As can be seen from Figure 8, all the modules have an input port sc_in_clk which provides uniform clock signals to all of them and 64-bit integer type input and output ports that can receive and send integers up to 64-bit respectively.

Other common feature among all the modules is that they all use the same type of threads – SC_CTHREAD(). All the modules have a function each defined in their threads which are run on the rising edge of the clock or the positive edge of the clock i.e. connect1() to connect7(). These functions have the necessary connections defined in them and conditions are written to send data to various modules at different points of time and similarly wait for other modules to send data and read them.

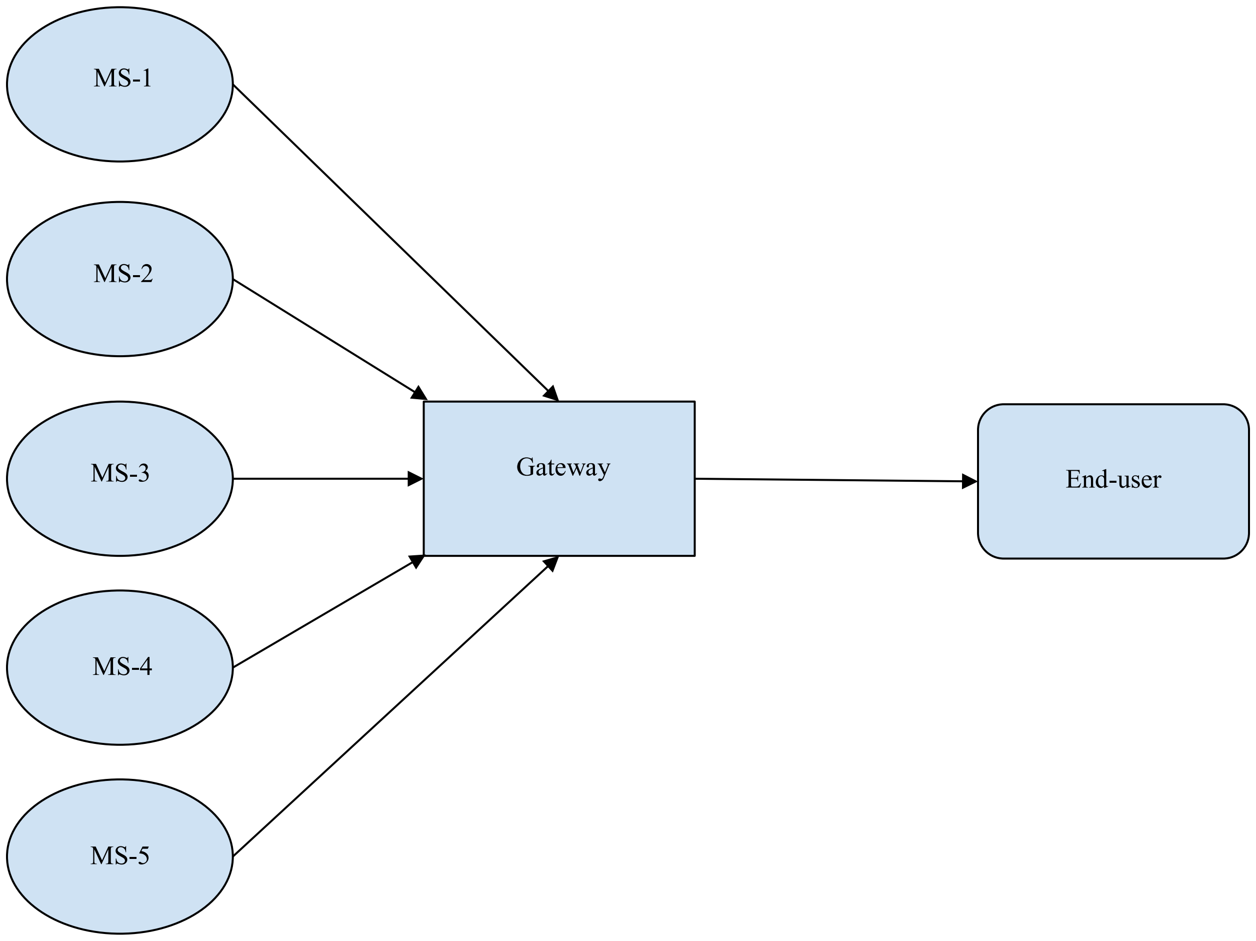

Figure 8. Simulator Architecture

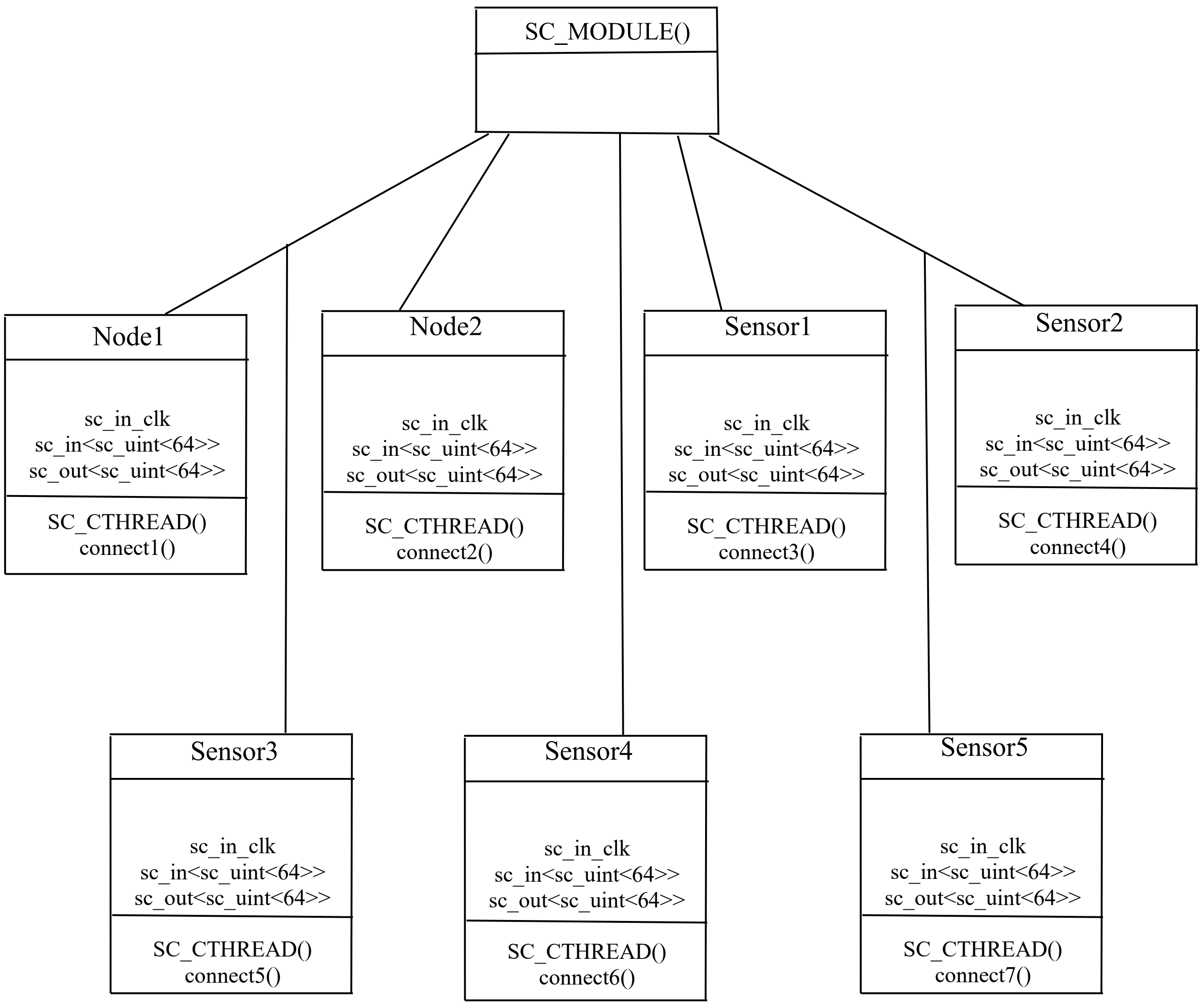

Figure 9. Class Diagram of the Simulator

The whole design is built from taking two modules as base and making necessary connections between the ports of those modules, similar to the example in section 3.6 and then expanding the design by adding one module each and adding signals to it which connect to already existing modules until all the seven modules are created and necessary connections are made. The gateway and the end-user are the modules that are created at the beginning and each sensor module is added. This is easier because the communication starts with a handshake between the gateway and the end-user. Also, since the sensors do not communicate with each other, it is easier to add each sensor module and add signals to them which connect to the gateway and the end-user (for session resumption).

The first bit of communication that takes place in the simulation is the certificate-based DTLS handshake between the smart e-health gateway and the end-user. There are six flights of messages in whole in the handshake, three sent by the gateway to the end-user and three sent by the end-user to the gateway. For the signals that are used to connect between any two ports from two different modules in SystemC, the data types that can be used for the signals are **bool** and **int**. But all of the parameters that are exchanged in the handshake take the form of character strings.

Therefore, we choose **int** as the data type of signals used here and convert each character of string into integer using its ASCII value. Also, the signals that are defined in this simulation can take a maximum of 64-bit integer values which is a 16-digit number. To make use of this, each character's ASCII value is subtracted by 55 to get the resultant value to a two-digit number. For example, the character "z's" ASCII value is 122 which is a three-digit number. But subtracting it with 55, which equals 67 is a two-digit number. Similarly, the character "A" whose ASCII value is 65 when subtracted by 55 will result in 10 which is the least two-digit number.

By doing this, in a 64-bit integer signal which is a 16-digit number, at any time a maximum of 8 characters can be sent through this signal. If a flight of messages needs more than 8 characters to be sent, then multiple signals are created to accommodate the remaining characters. Once a module converts the string into integers by the method mentioned above, the message is sent. The module which receives the said message converts back the integers obtained into strings and prints the messages received.

This is done enough times to send and receive all the messages in the handshake. If-else conditions are used to check if at any point, the message that needs to be received either by the gateway or the end-user is what it should be. If the value received by a module is not what it is supposed to be, then the handshake is terminated and the simulation is stopped.

The simulation starts when the user enters "Begin" to the screen. Then the first message sent by the end-user is "Hello Params", where Hello is Client Hello and Params are the security parameters the end-user sends to the gateway in the actual handshake. As explained, "Hello Params" is not actually sent but the integer 1746535356 is sent for "Hello" and 254259425460 is sent for the word “Params”. (The ASCII values of ‘H’, ‘e’, ‘l’, ‘l’, ‘o’ are 72, 101, 108, 108 and 111 and when each of them is subtracted by 55, it will result in 17, 46, 53, 53 and 56. All these values are concatenated and sent as one value. Refer to the Appendix – Node1.cpp). The gateway responds with the same message with its security parameters which are used in the actual handshake. The next message “Cookie” is sent by the end-user to the gateway which signifies an additional cookie that is sent by the client.

The next flight of messages by the gateway is “Hello ServerCertificates KeyExchange CertificateRequest Finish”. In the actual handshake, these messages represent the negotiated encryption method in Hello, the certificates of the gateway in ServerCertificates, other parameters that are required to calculate the master secret key in KeyExchange and the list of certificates that are valid in the gateway in the CertificateRequest and finally a Finish message.

The end-user responds with “ClientCertificates KeyExchange CertificateVerify MasterSecret”. The end-user sends its certificate in ClientCertificate, additional requirements to compute the master secret key are included in the KeyExchange and CertificateVerify has the proof that the end-user has the parameters which are needed to compute the private key. MasterSecret has the pre-master secret key. The gateway also sends the MasterSecret and a SessionTicket to the end-user.

Finally, “ChangeCipherSpec” and Finished messages are exchanged between the end-user and the gateway to signify that the next messages are sent and received in an encrypted format which was agreed beforehand. “Finished” concludes the DTLS handshake.

Once the handshake is successful, the DTLS session resumption needs to be simulated. A sensor module – Sensor1 is created and signals are established which connect the sensor module to the end-user module and the gateway module. Sensor1 needs to wait for a few clock cycles until it can start communicating with the end-user. The concept of wait() and how it works in the SystemC simulator is explained in the section 5.3. In the simulation, the authentication of sensors through session resumption takes place simultaneously. First, Sensor1 exchanges messages with the end-user and the smart e-health gateway before other sensors are created and connected to end-user and gateway modules. The gateway sends a message to the sensor, “Enduser Authorized” meaning the end-user that the gateway was performing the handshake with is authorized. Then, the end-user sends a “Client Hello” message which actually has the session resumption extension with a session ticket. The sensor responds with “ServerHello SessionTicket MasterSecret ChangeCipherSpec Finished”. The ServerHello has an empty session resumption extension, SessionTicket has the new session ticket credentials, MasterSecret has the information about master secret. ChangeCipherSpec has the information about the encrypting algorithm and Finished secures the channel. The end-user also responds with a ChangeCipherSpec and Finished messages.

The same session resumption technique is extended to Sensors 2, 3, 4 and 5. The same messages are exchanged between the sensor modules and end-user modules to authenticate all the sensors. Then, all the modules are ready to send and receive data.

4.8 Data flow from a file

Once the authentication and authorization processes for sensors and the gateway are done, each sensor reads data which are in integer format from a .txt file and relays each reading to the gateway. These numbers can be assumed to be readings obtained from any physical medical sensor. The relaying of these numbers is done by ports and signals. Signals are connected between each sensor and the gateway and from the gateway to the end-user to send data. Once the gateway module reads the numbers, it transfers them to the end-user module which prints each number on the screen.

After a value from a sensor (say, sensor1) is output on the screen, the next sensor (sensor2) sends its reading to the gateway. So, while a sensor sends its reading, the other sensors need to wait for a certain period of time until they can send their values to the gateway. For example, sensor1 has to wait until sensors 2, 3, 4 and 5 send a value each to the end-user through the gateway. Similarly, the end-user has to wait while a sensor sends a reading to the gateway and the gateway has to wait while the reading is output on the screen. Depending on the number of values to be sent, the simulation time can be changed. In the simulation, each text file for each sensor had five values and the simulation ran for a total of 200 nanoseconds where the period of the clock was 2 ns.

Chapter 5

Results

The simulator can be used for automating secure end-to-end communication between medical sensors and end-users. Although the preliminary design had five medical sensors and one gateway along with one end-user, a similar design can be implemented with a different number of modules and the format of the data that is sent and received can also be varied.

5.1 Where can this simulation be used?

The authentication and authorization methods that are demonstrated for the simulator can be replicated in not just healthcare IoT, but in other domains where secure communication between a resource constrained device and a remote server needs to be shown. The prerequisites for simulating the design is the system needs to have an operating system which can compile and run SystemC programs.

Since the design of the simulator depends on the number of modules, ports that each module possesses and the signals that are used to connect between the ports of two different modules, once the simulator is ready, it is not possible to change the number of sensors or gateways without altering the SystemC code. Therefore, it is possible to add or remove the components of the design or scale the simulator only by changing the code and not in any other way.

The datatypes used to define signals which are used to send and receive data are of the type **int**. Therefore, the data that can be sent by the sensors and received by the end-user need to be integers. Other datatypes can also be accepted by the end-user provided there is a function in the SystemC code which can convert character strings or just characters into integers, but that is only used to perform certificate-based DTLS handshake and session resumption technique, but not data transfer. Hence, only integers can be sent.

There is only one signal that is connected between each sensor and the gateway for the secure data transfer and one signal that is connected between the gateway and the end-user for the data transfer. Each signal can accommodate an integer that is less than or equal to a 64-bit integer. Therefore, the data that needs to be sent cannot exceed this value. If there is a requirement to send a value higher than a 64-bit integer, then multiple signals need to be connected between the modules. The number of signals between the sensors and the gateway should equal the number of signals between the gateway and the end-user for the data transfer.

5.2 Simulator testing and verification

As mentioned in Section 5.1, this simulator can be used to represent any kind of secure communication between a resource constrained device and an end-user with the help of a gateway or an intermediate device. There are many applications of IoT where resource constrained devices and sensors are used and the data that is read by the sensors needs to be sent regularly in order to get timely updates at the receiver's end. A few examples include, a low-energy location tracker using GPS technology, humidity and temperature sensor, fire and smoke detector, sensors which monitor air quality and pollution, other tracking sensors used to calculate the number of products manufactured, products delivered and stored in various inventories.

Since the data that can be sent and received is only in the integer format, the above sensors can only send and receive a number like temperature, location in coordinates, amount of carbon dioxide present in the air and number of items available. For fire and smoke detector, a boolean value can be used to convey if fire and smoke is detected or not. For example, 1 for fire and smoke and 0 for no fire and no smoke.

This simulator can also be used just to transfer data between three parties which don't have to be components in an IoT environment. The simulator can be used to demonstrate how to secure a channel between two levels of hierarchy. Like, securing a communication channel between multiple employees at the same level and sending messages to their manager.

5.3 Case Studies

To calculate the number of clock cycles or the amount of time it takes in SystemC for a set of five readings to be read at the end-user from each sensor, there is a need to understand how the command **wait()** works in the simulation.

When the program is run, the user must enter ‘Begin’ to start the simulation. Then, the end-user sends a Hello message to the gateway which takes place in a single clock cycle. The gateway receives the ‘Hello’ message from the server and sends a ‘Hello’ message of its own to the end-user. This happens after one clock cycle, since it takes one clock cycle to perform an action (all the instructions are run on the positive edge of the clock) the gateway has to wait for a single clock cycle (or **wait(1)**) before it can read the received message and send its first message.

The end-user waits for two clock cycles between sending its first message and receiving the first message from the gateway and responding with another message in the certificate-based DTLS handshake. Similarly, the gateway waits for two clock cycles when the end-user reads and responds with a message. In the handshake that is implemented, there are four messages each sent by the end-user and the gateway and finally the end-user receives the last message sent by gateway. Each message takes one clock-cycle, therefore the total number of clock cycles it takes to complete the handshake in the simulation is 9 clock cycles.

In the 10th clock cycle the gateway sends a message, “Enduser Authorized”, to the first sensor. Therefore, Sensor1 has to wait for 10 clock cycles to receive the message from the gateway. In the session resumption, there are six steps that take place. The total number of clock cycles it takes to finish the session resumption of Sensor1 is 6, and hence the total number of clock cycles that take place until this point are 9 + 6 = 15.

Similarly, for the session resumption of other four sensors, it takes 4*6 = 24 clock cycles. The total number of clock cycles it takes to reach the beginning of sending the first reading from Sensor1 is 15 + 24 = 39.

While the session resumption of Sensors2 is taking place, Sensors1, 3, 4, 5, the gateway and the end-user are in wait state. Likewise, when session resumption of Sensor3 is taking place, Sensors1, 2, 4, 5, the gateway and the end-user are all in wait state and so on.

For each sensor to send data to the gateway takes one clock cycle and for the gateway to send the same data to the end-user takes another clock cycle. It also takes one clock cycle for the end-user to display the output it received. Therefore, for each reading three clock cycles take place. For five readings, it takes a total of 15 clock cycles and for five sensors, it takes 15*5 = 75 clock cycles. The total number of clock cycles it takes for the whole simulation is 39 + 75 = 114 clock cycles. In this simulation, the time units mentioned is nano-seconds and the period of the clock is 2 ns. The total time taken is therefore, 114*2 = 228 ns.

For each extra reading, there are three clock cycles that are added to the total simulation time. The total time assuming that the period of the clock is 2ns, can be calculated by $(39*2 + 3*(r_1 + r_2 + r_3 + r_4 + r_5))$ where r_1, r_2, r_3, r_4 and r_5 are the number of readings of Sensors1, 2, 3, 4 and 5 respectively that are supposed to be read at the end-user. Accordingly, the total amount of time that the simulation must run has to be adjusted so that all the readings can be read at the end-user. The summarization of the amount of time it takes on SystemC is shown in Table 1.

For the readings in Table 1, the assumptions taken are the same as those that are applied in the simulator i.e. the simulator has one gateway and one end-user, the time units chosen is nano-seconds, the period of the clock is 2ns and the duty cycle is 50%. The number of readings per sensor and the number of sensors are chosen randomly.

Number of Sensors	**Number of readings/sensor**	**Minimum SystemC time needed (ns)**
5	5	228
5	10	378
5	15	528
4	10	306
4	6	210
3	8	198
2	12	186

Table 1. Time taken by SystemC clock depending on the number of sensors and readings per sensor.

Chapter 6

Conclusion

In this thesis, a simulator was designed using SystemC which highlights all the steps involved in a secure end-to-end data transfer in healthcare IoT. The certificate-based DTLS handshake was simulated first between the end-user and the gateway which is connected to the sensors, session resumption technique for securing end-to-end communication. Finally, the simulation also showed how data transfer takes place between medical sensor modules and the end-user module.

Although this thesis covered many important aspects of the target system as specified in the work by Moosavi et al. [7], some of the elements which were beyond the scope of the thesis were not implemented. In the target system, the concept of secure data transfer extended further to include mobility of the patients on whom sensors are attached so that they don't have to be confined to a single spot. The sensors connect to the closest smart e-health gateway to them which can transfer the data to the end-user. Implementing this to the simulation can be tricky because sensors connecting to different gateways mean that the signals differ in SystemC and the path of the transfer of data is not constant. But this can be implemented if there is a model of a patient's sensors connecting to different gateways at various points of time.

Also, while simulating DTLS handshake and the session resumption, what all parameters are transferred are shown but the actual messages are not shown. For example, in DTLS handshake, one of the messages sent by the end-user is ClientCertificate which is shown in the simulation but how a message which has client certificate is not shown. This can be improved to other messages exchanged and the actual data transfer that takes place can also be done in encrypted format where the encryption algorithm is agreed by both the end-user and the gateway and the decryption can be done at the receiving end.

The time it takes to simulate in SystemC is very reasonable and highlighting all the details of the simulation can be very educative on how secure data transfers can take place in a healthcare IoT environment.

References

[1] Louis Columbus. “10 Charts That Will Challenge Your Perspective of IoT’s Growth.” Internet: https://www.forbes.com/sites/louiscolumbus/2018/06/06/10-charts-that-will-challenge-your-perspective-of-iots-growth/#76870bb3ecce, June 6, 2018 [Accessed March 5, 2019].

[2] Symantec. “Internet Security Threat Report ISTR”. Vol. 23. Available: https://www.symantec.com/content/dam/symantec/docs/reports/istr-23-2018-en.pdf [Accessed March 5, 2019].

[3] Josh Fruhlinger. “The Mirai botnet explained: How teen scammers and CCTV cameras almost brought down the internet.” Internet: https://www.csoonline.com/article/3258748/the-mirai-botnet-explained-how-teen-scammers-and-cctv-cameras-almost-brought-down-the-internet.html, March 9, 2018 [Accessed March 5, 2019].

[4] Corero. “Mirai Botnet DdoS Attack Type,” Internet: https://www.corero.com/resources/ddos-attack-types/mirai-botnet-ddos-attack.html, [Accessed March 5, 2019].

[5] Kaspersky Lab. “New IoT-malware grew three-fold in H1 2018.” Internet: https://www.kaspersky.com/about/press-releases/2018_new-iot-malware-grew-three-fold-in-h1-2018, Sept. 18, 2018 [Accessed March 5, 2019].

[6] Sanaz Rahimi Moosavi, Tuan Nguyen Gia, Amir-Mohammad Rahmani, Ethiopia Nigussie, Seppo Virtanen, Jouni Isoaho and Hannu Tenhunen. “A Secure and Efficient Authentication and Authorization Architecture for IoT-Based Healthcare Using Smart Gateways,” in *6th International Conference on Ambient Systems, Networks and Technologies, London, UK* vol. 52, pp. 452-459, 2015.

[7] Sanaz Rahimi Moosavi, Tuan Nguyen Gia, Ethiopia Nigussie, Amir M. Rahmani, Seppo Virtanen, Hannu Tenhunen and Jouni Isoaho. “End-to-end security scheme for mobility enabled healthcare Internet of Things,” in *Future Generation Computer Systems,* vol. 644, pp. 108-124

[8] David C. Black and Jack Donovan. *SystemC: From the Ground Up*. Boston, MA: Kluwer Academic Publishers, 2004.

[9] Shancang Li and Li Da Xu. “The Internet of Things: A Survey,” in *Information Systems Frontiers,* vol. 17, pp. 243-259, April 2015.

[10] Michael Weyrich and Christof Ebert. “Reference Architectures for the Internet of Things.” *IEEE Software* vol. 33, pp. 112-116, Jan, 2016.

[11] Andrew Whitmore, Anurag Agarwal and Li Da Xu. “The Internet of Things – A Survey of Topics and Trends,” in *Information Systems Frontiers,* vol. 17, pp. 261-274, 2015.

[12] Saurabh Singh, Pradip Kumar Sharma, Seo Yeon Moon and Jong Hyuk Park. “Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions,” in *Journal of Ambient Intelligence and Humanized Computing,* vol. 8, pp. 1-18, April 2017.

[13] Debasis Bandyopadhyay and Jaydip Sen. “Internet of Things: Applications and Challenges in Technology and Standardization,” in *Wireless Personal Communications,* vol. 58, pp. 49-69, May 2011.

[14] Andrew Meola. “Automotive Industry Trends: IoT Connected Smart Cars and Vehicles.” Internet: https://www.businessinsider.com/internet-of-things-connected-smart-cars-2016-10?r=US&IR=T&IR=T, Dec. 20, 2016 [Accessed March 15, 2019].

[15] Claudio Rosmino and Robert Hackwill. “5G, blockchain and IoT: what does the future of telecommunications look like?” Internet: https://www.euronews.com/2018/10/31/5g-blockchain-and-iot-the-future-of-telecommunication, Nov. 5, 2018 [Accessed March 15, 2019].

[16] Insights team. “Logistics 4.0: How IoT is Transforming The Supply Chain.” Internet: https://www.forbes.com/sites/insights-inteliot/2018/06/14/logistics-4-0-how-iot-is-transforming-the-supply-chain/#2f87f663880f, Jun 14, 2018 [Accessed March 20, 2019].

[17] IoT For All. “Where IoT Meets the Environment: Building a Greener Future.” Internet: https://www.iotforall.com/iot-environment-greener-future/, Jan. 8, 2019. [Accessed March 15, 2019].

[18] Anna Gerber. “Top 10 IoT Security Challenges.” Internet: https://developer.ibm.com/articles/iot-top-10-iot-security-challenges/, Nov. 17, 2017 [Accessed March. 18, 2019].

[19] Reda Chouffani. “Current and future applications of IoT in healthcare.” Internet: https://internetofthingsagenda.techtarget.com/feature/Can-we-expect-the-Internet-of-Things-in-healthcare, [Accessed March 18, 2019].

[20] Phillip A. Laplante and Nancy Laplante. “The Internet of Things in Healthcare: Potential Applications and Challenges” in *IT Professional* vol. 18, pp. 2-4, May 2016.

[21] Dr. Rajashekhar Karjagi and Manish Jindal. “What can IoT do for healthcare?” Internet: https://www.wipro.com/en-IN/business-process/what-can-iot-do-for-healthcare-/, [Accessed March 21, 2019].

[22] Nasrullah Patel. “Internet of Things in Healthcare: Applications, benefits, and challenges.” Internet: https://www.peerbits.com/blog/internet-of-things-healthcare-applications-benefits-and-challenges.html, [Accessed March 21, 2019].

[23] “Transforming care with Digital Medicine.” Internet: https://www.proteus.com/evidence, [Accessed March 21, 2019].

[24] “Using the nervous system to change medicine.” Internet: https://www.thync.com, [Accessed March 21, 2019].

[25] Wassnaa Al-mawee. “Privacy and Security Issues in IoT Healthcare Applications for the Disabled Users a Survey.” Master’s Thesis, Western Michigan University, USA, 2012.

[26] Somasundaram Ragupathy and Mythili Thirugnanam. “IoT in Healthcare: Breaching Security Issues.” in *Advances in Information Security, Privacy, and Ethics*, Feb. 2017, pp. 174-188.

[27] Blackberry. “Securing the Internet of Healthcare Things.” Internet: https://global.blackberry.com/content/dam/blackberry-com/asset/enterprise/pdf/wp-cybersecurity-healthcare.pdf, Nov. 2017 [Accessed March 25, 2019].

[28] Bill Siwicki. “71% of IoT Medical device ransomware infections caused by user practice issues.” Internet: https://www.healthcareitnews.com/news/71-iot-medical-device-ransomware-infections-caused-user-practice-issues, March 5, 2018 [Accessed March 25, 2019].

[29] Selena Larson. "FDA confirms that St. Jude's cardiac devices can be hacked" Internet: https://money.cnn.com/2017/01/09/technology/fda-st-jude-cardiac-hack/, Jan. 9, 2017. [Accessed March 28, 2019].

[30] Guest Writer. "The 5 worst examples of IoT hacking and vulnerabilities in recorded history." Internet: https://www.iotforall.com/5-worst-iot-hacking-vulnerabilities, May 10, 2017. [Accessed March 28, 2019].

[31] Preston Gralla. "Medical IoT devices: the security nightmare that keeps CIOs up late at night." Internet: https://www.hpe.com/us/en/insights/articles/medical-iot-devices-the-security-nightmare-that-keeps-cios-up-late-at-night-1709.html, Sep. 25, 2017. [Accessed March 29, 2019].

[32] Frank Ghenassia and Alain Clouard. "TLM: An Overview and Brief History" in *Transaction Level Modeling with SystemC.* Frank Ghenassia, Springer, Boston, MA.

[33] "SystemC". Internet: https://accellera.org/downloads/standards/systemc, [Accessed March 31, 2019].

[34] Accellera. "Install notes for SystemC release 2.3". Internet: https://github.com/systemc/systemc-2.3/blob/master/INSTALL, March 24, 2012. [Accessed April 2, 2019].

[35] Cristian Slav. "Learning SystemC: #001 Data Types". Internet: http://cfs-vision.com/2017/08/14/learning-systemc-001-data-types/#sc_logic, Aug. 14, 2017. [Accessed April 2, 2019].

[36] Greg Cline. "IoT and Analytics: Better Manufacturing Decisions in the era of Industry 4.0". Internet: https://www.ibm.com/downloads/cas/ZPB2PN2G, Aug. 2017 [Accessed April 3, 2019].

Appendix A

main.cpp

```
#include <systemc.h>
#include "Node1.h"
#include "Node2.h"
#include "sensor1.h"
#include "sensor2.h"
#include "sensor3.h"
#include "sensor4.h"
#include "sensor5.h"

int sc_main (int ac, char* av[])
{
      sc_report_handler::set_actions("/IEEE_Std_1666/deprecated",
SC_DO_NOTHING);
      sc_signal<sc_uint<64>> s1("Signal-1");
      sc_signal<sc_uint<64>> s2("Signal-2");
      sc_signal<sc_uint<64>> s3("Signal-3");
      sc_signal<sc_uint<64>> s4("Signal-4");
      sc_signal<sc_uint<64>> s5("Signal-5");
      sc_signal<sc_uint<64>> s6("Signal-6");
      sc_signal<sc_uint<64>> s7("Signal-7");
      sc_signal<sc_uint<64>> s8("Signal-8");
      sc_signal<sc_uint<64>> s9("Signal-9");
      sc_signal<sc_uint<64>> s10("Signal-10");
      sc_signal<sc_uint<64>> s11("Signal-11");
      sc_signal<sc_uint<64>> s12("Signal-12");
      sc_signal<sc_uint<64>> s13("Signal-13");
      sc_signal<sc_uint<64>> s14("Signal-14");
      sc_signal<sc_uint<64>> s15("Signal-15");
      sc_signal<sc_uint<64>> s16("Signal-16");
      sc_signal<sc_uint<64>> s17("Signal-17");
      sc_signal<sc_uint<64>> s18("Signal-18");
      sc_signal<sc_uint<64>> s19("Signal-19");
      sc_signal<sc_uint<64>> s20("Signal-20");
      sc_signal<sc_uint<64>> s21("Signal-21");
      sc_signal<sc_uint<64>> s22("Signal-22");
      sc_signal<sc_uint<64>> s23("Signal-23");
      sc_signal<sc_uint<64>> s24("Signal-24");
      sc_signal<sc_uint<64>> s25("Signal-25");
      sc_signal<sc_uint<64>> s26("Signal-26");
      sc_signal<sc_uint<64>> s27("Signal-27");
      sc_signal<sc_uint<64>> s28("Signal-28");
      sc_signal<sc_uint<64>> s29("Signal-29");
      sc_signal<sc_uint<64>> s30("Signal-30");
      sc_signal<sc_uint<64>> s31("Signal-31");
      sc_signal<sc_uint<64>> s32("Signal-32");
      sc_signal<sc_uint<64>> s33("Signal-33");
```

```
sc_signal<sc_uint<64>> s34("Signal-34");
sc_signal<sc_uint<64>> s35("Signal-35");
sc_signal<sc_uint<64>> s36("Signal-36");
sc_signal<sc_uint<64>> s37("Signal-37");
sc_signal<sc_uint<64>> s38("Signal-38");
sc_signal<sc_uint<64>> s39("Signal-39");
sc_signal<sc_uint<64>> s40("Signal-40");
sc_signal<sc_uint<64>> s41("Signal-41");
sc_signal<sc_uint<64>> s42("Signal-42");
sc_signal<sc_uint<64>> s43("Signal-43");
sc_signal<sc_uint<64>> s44("Signal-44");
sc_signal<sc_uint<64>> s45("Signal-45");
sc_signal<sc_uint<64>> s46("Signal-46");
sc_signal<sc_uint<64>> s47("Signal-47");
sc_signal<sc_uint<64>> s48("Signal-48");
sc_signal<sc_uint<64>> s49("Signal-49");
sc_signal<sc_uint<64>> s50("Signal-50");
sc_signal<sc_uint<64>> s51("Signal-51");
sc_signal<sc_uint<64>> s52("Signal-52");
sc_signal<sc_uint<64>> s53("Signal-53");
sc_signal<sc_uint<64>> s54("Signal-54");
sc_signal<sc_uint<64>> s55("Signal-55");
sc_signal<sc_uint<64>> s56("Signal-56");
sc_signal<sc_uint<64>> s57("Signal-57");
sc_signal<sc_uint<64>> s58("Signal-58");
sc_signal<sc_uint<64>> s59("Signal-59");
sc_signal<sc_uint<64>> s60("Signal-60");
sc_signal<sc_uint<64>> s61("Signal-61");
sc_signal<sc_uint<64>> s62("Signal-62");
sc_signal<sc_uint<64>> s63("Signal-63");
sc_signal<sc_uint<64>> s64("Signal-64");
sc_signal<sc_uint<64>> s65("Signal-65");
sc_signal<sc_uint<64>> s66("Signal-66");
sc_signal<sc_uint<64>> s67("Signal-67");
sc_signal<sc_uint<64>> s68("Signal-68");
sc_signal<sc_uint<64>> s69("Signal-69");
sc_signal<sc_uint<64>> s70("Signal-70");
sc_signal<sc_uint<64>> s71("Signal-71");
sc_signal<sc_uint<64>> s72("Signal-72");
sc_signal<sc_uint<64>> s73("Signal-73");
sc_signal<sc_uint<64>> s74("Signal-74");
sc_signal<sc_uint<64>> s75("Signal-75");
sc_signal<sc_uint<64>> s76("Signal-76");
sc_signal<sc_uint<64>> s77("Signal-77");
sc_signal<sc_uint<64>> s78("Signal-78");
sc_signal<sc_uint<64>> s79("Signal-79");
sc_signal<sc_uint<64>> s80("Signal-80");
sc_signal<sc_uint<64>> s81("Signal-81");
sc_signal<sc_uint<64>> s82("Signal-82");
sc_signal<sc_uint<64>> s83("Signal-83");
sc_signal<sc_uint<64>> s84("Signal-84");
sc_signal<sc_uint<64>> s85("Signal-85");
```

```
sc_signal<sc_uint<64>> s86("Signal-86");
sc_signal<sc_uint<64>> s87("Signal-87");

sc_uint<64> final = 0;
char message1[6];
int message2[6];
int temp;
int i;

sc_clock clock("Clock", 2, 0.5, 0.0);

node_1 n1 ("n1");
n1.clk(clock);
n1.a(s1);
n1.b(s2);
n1.c(s3);
n1.d(s4);
n1.e(s5);
n1.f(s6);
n1.g(s7);
n1.h(s8);
n1.i(s9);
n1.j(s10);
n1.k(s11);
n1.l(s12);
n1.m(s16);
n1.n(s17);
n1.o(s18);
n1.p(s19);
n1.q(s20);
n1.r(s21);
n1.s(s22);
n1.t(s23);
n1.u(s24);
n1.v(s25);
n1.w(s26);
n1.x(s30);
n1.y(s31);
n1.z(s32);
n1.aa(s33);
n1.ab(s34);
n1.ac(s35);
n1.ad(s36);
n1.ae(s37);
n1.af(s38);
n1.ag(s39);
n1.ah(s40);
n1.ai(s44);
n1.aj(s45);
n1.ak(s46);
n1.al(s47);
n1.am(s48);
```

```
n1.an(s49);
n1.ao(s50);
n1.ap(s51);
n1.aq(s52);
n1.ar(s53);
n1.as(s54);
n1.at(s58);
n1.au(s59);
n1.av(s60);
n1.aw(s61);
n1.ax(s62);
n1.ay(s63);
n1.az(s64);
n1.ba(s65);
n1.bb(s66);
n1.bc(s67);
n1.bd(s68);
n1.be(s72);
n1.bf(s73);
n1.bg(s74);
n1.bh(s75);
n1.bi(s76);
n1.bj(s77);
n1.bk(s78);
n1.bl(s79);
n1.bm(s80);
n1.bn(s81);
n1.bo(s82);

node_2 n2 ("n2");
n2.clk(clock);
n2.a(s2);
n2.b(s1);
n2.c(s3);
n2.d(s4);
n2.e(s5);
n2.f(s6);
n2.g(s7);
n2.h(s8);
n2.i(s9);
n2.j(s10);
n2.k(s11);
n2.l(s12);
n2.n(s13);
n2.o(s14);
n2.p(s15);
n2.q(s27);
n2.r(s28);
n2.s(s29);
n2.t(s41);
n2.u(s42);
n2.v(s43);
```

```
n2.w(s55);
n2.x(s56);
n2.y(s57);
n2.z(s69);
n2.aa(s70);
n2.ab(s71);
n2.ac(s83);
n2.ad(s84);
n2.ae(s85);
n2.af(s86);
n2.ag(s87);

sensor_1 s_1 ("s_1");
s_1.clk(clock);
s_1.a(s13);
s_1.b(s14);
s_1.c(s15);
s_1.d(s16);
s_1.e(s17);
s_1.f(s18);
s_1.g(s19);
s_1.h(s20);
s_1.i(s21);
s_1.j(s22);
s_1.k(s23);
s_1.l(s24);
s_1.m(s25);
s_1.n(s26);
s_1.o(s83);

sensor_2 s_2("s_2");
s_2.clk(clock);
s_2.a(s27);
s_2.b(s28);
s_2.c(s29);
s_2.d(s30);
s_2.e(s31);
s_2.f(s32);
s_2.g(s33);
s_2.h(s34);
s_2.i(s35);
s_2.j(s36);
s_2.k(s37);
s_2.l(s38);
s_2.m(s39);
s_2.n(s40);
s_2.o(s84);

sensor_3 s_3("s_3");
s_3.clk(clock);
s_3.a(s41);
s_3.b(s42);
```

```
s_3.c(s43);
s_3.d(s44);
s_3.e(s45);
s_3.f(s46);
s_3.g(s47);
s_3.h(s48);
s_3.i(s49);
s_3.j(s50);
s_3.k(s51);
s_3.l(s52);
s_3.m(s53);
s_3.n(s54);
s_3.o(s85);

sensor_4 s_4("s_4");
s_4.clk(clock);
s_4.a(s55);
s_4.b(s56);
s_4.c(s57);
s_4.d(s58);
s_4.e(s59);
s_4.f(s60);
s_4.g(s61);
s_4.h(s62);
s_4.i(s63);
s_4.j(s64);
s_4.k(s65);
s_4.l(s66);
s_4.m(s67);
s_4.n(s68);
s_4.o(s86);

sensor_5 s_5("s_5");
s_5.clk(clock);
s_5.a(s69);
s_5.b(s70);
s_5.c(s71);
s_5.d(s72);
s_5.e(s73);
s_5.f(s74);
s_5.g(s75);
s_5.h(s76);
s_5.i(s77);
s_5.j(s78);
s_5.k(s79);
s_5.l(s80);
s_5.m(s81);
s_5.n(s82);
s_5.o(s87);

cout << "Enter 'Begin' to begin the simulation: " << endl;
```

```
	for (i = 0; i < 5; i++)
	{
		cin >> message1[i];
	}

	for (i = 0; i < 5; i++)
	{
		message2[i] = int(message1[i]);
	}

	for (i = 0; i < 5; i++)
	{
		temp = 0;
		temp = ((message2[i]-55)* pow(10, (2*4 - (2*i))));
		final = temp + final;
	}

	s1.write(final);

	sc_start(200, SC_NS);

	return 0;
}
```

Node1.h

```
SC_MODULE(node_1)
{
	sc_in_clk clk;
	sc_in<sc_uint<64>> a;
	sc_out<sc_uint<64>> b;
	sc_out<sc_uint<64>> c;
	sc_in<sc_uint<64>> d;
	sc_in<sc_uint<64>> e;
	sc_in<sc_uint<64>> f;
	sc_in<sc_uint<64>> g;
	sc_out<sc_uint<64>> h;
	sc_out<sc_uint<64>> i;
	sc_out<sc_uint<64>> j;
	sc_out<sc_uint<64>> k;
	sc_out<sc_uint<64>> l;
	sc_out<sc_uint<64>> m;
	sc_out<sc_uint<64>> n;
	sc_in<sc_uint<64>> o;
	sc_in<sc_uint<64>> p;
	sc_in<sc_uint<64>> q;
	sc_in<sc_uint<64>> r;
	sc_in<sc_uint<64>> s;
	sc_in<sc_uint<64>> t;
```

```
sc_in<sc_uint<64>> u;
sc_out<sc_uint<64>> v;
sc_out<sc_uint<64>> w;
sc_out<sc_uint<64>> x;
sc_out<sc_uint<64>> y;
sc_in<sc_uint<64>> z;
sc_in<sc_uint<64>> aa;
sc_in<sc_uint<64>> ab;
sc_in<sc_uint<64>> ac;
sc_in<sc_uint<64>> ad;
sc_in<sc_uint<64>> ae;
sc_in<sc_uint<64>> af;
sc_out<sc_uint<64>> ag;
sc_out<sc_uint<64>> ah;
sc_out<sc_uint<64>> ai;
sc_out<sc_uint<64>> aj;
sc_in<sc_uint<64>> ak;
sc_in<sc_uint<64>> al;
sc_in<sc_uint<64>> am;
sc_in<sc_uint<64>> an;
sc_in<sc_uint<64>> ao;
sc_in<sc_uint<64>> ap;
sc_in<sc_uint<64>> aq;
sc_out<sc_uint<64>> ar;
sc_out<sc_uint<64>> as;
sc_out<sc_uint<64>> at;
sc_out<sc_uint<64>> au;
sc_in<sc_uint<64>> av;
sc_in<sc_uint<64>> aw;
sc_in<sc_uint<64>> ax;
sc_in<sc_uint<64>> ay;
sc_in<sc_uint<64>> az;
sc_in<sc_uint<64>> ba;
sc_in<sc_uint<64>> bb;
sc_out<sc_uint<64>> bc;
sc_out<sc_uint<64>> bd;
sc_out<sc_uint<64>> be;
sc_out<sc_uint<64>> bf;
sc_in<sc_uint<64>> bg;
sc_in<sc_uint<64>> bh;
sc_in<sc_uint<64>> bi;
sc_in<sc_uint<64>> bj;
sc_in<sc_uint<64>> bk;
sc_in<sc_uint<64>> bl;
sc_in<sc_uint<64>> bm;
sc_out<sc_uint<64>> bn;
sc_out<sc_uint<64>> bo;

SC_CTOR(node_1)
{
      SC_CTHREAD(connect1, clk.pos());
}
```

```
      void connect1();
}
```

Node1.cpp

```
#include <systemc.h>
#include "Node1.h"

void node_1::connect1()
{
      while(true)
      {
            sc_uint<64> final;
            sc_uint<64> write1;
            sc_uint<64> read0;
            read0 = a.read();
            int temp, temp2, temp3, temp4, temp5;
            int no;
            char message1[7], message3[6], message5[7], message7[6],
message9[6], message11[8], message13[7], message15[9], message17[9],
message19[9], message21[9], message23[9];
            int message2[7], message4[6], message6[7], message8[6],
message10[6], message12[8], message14[7], message16[9], message18[9],
message20[9], message22[9], message24[9];

            if (read0 == 1146485055)
            {
                  cout << "Client message 1: Hello Params" << endl;
                  final = 1746535356;
                  write1 = 254259425460;
                  b.write(final);
                  c.write(write1);

                  wait(2);

                  sc_uint<64> read1;
                  temp = 0;
                  temp2 = 0;
                  read0 = a.read();
                  read1 = d.read();

                  if (read0 == 1746535356 && read1 == 254259425460)
                  {
                        for (no = 4; no >= 0; no--)
                        {
                              temp = read0 % 100;
                              read0 = read0 / 100;
                              message8[no] = (temp + 55);
```

```
                    }

                    for (no = 0; no < 5; no++)
                    {
                         message7[no] = char(message8[no]);
                    }

                    cout << "Client receives: ";

                    for (no = 0; no < 5; no++)
                    {
                         cout << message7[no];
                    }
                    cout << " ";

                    for (no = 5; no >= 0; no--)
                    {
                         temp2 = read1 % 100;
                         read1 = read1 / 100;
                         message2[no] = (temp2 + 55);
                    }

                    for (no = 0; no < 6; no++)
                    {
                         message1[no] = char(message2[no]);
                    }

                    for (no = 0; no < 6; no++)
                    {
                         cout << message1[no];
                    }
                    cout << endl;

                    cout << "Client message 2: Cookie" << endl;
                    write1 = 125656525046;
                    b.write(write1);

                    wait(2);

                    sc_uint<64> read2, read3, read4, read5;
                    sc_uint<64> write0, write2, write3, write4,
write5, write6, write7;
                    temp = temp2 = temp3 = temp4 = temp5 = 0;
                    read1 = a.read();
                    read2 = d.read();
                    read3 = e.read();
                    read4 = f.read();
                    read5 = g.read();
                    if (read1 == 1746535356 && read2 == 281246596160
&& read3 == 2046661465 && read4 == 12465961274658 && read5 ==
155055506049)
                    {
```

```
for (no = 4; no >= 0; no--)
{
      temp = read1 % 100;
      read1 = read1 / 100;
      message4[no] = (temp + 55);
}
for (no = 0; no < 5; no++)
{
      message3[no] = char(message4[no]);
}
cout << "Client receives: ";
for (no = 0; no < 5; no++)
{
      cout << message3[no];
}
cout << " ";

for (no = 5; no >= 0; no--)
{
      temp2 = read2 % 100;
      read2 = read2 / 100;
      message6[no] = (temp2 + 55);
}
for (no = 0; no < 6; no++)
{
      message5[no] = char(message6[no]);
}
for (no = 0; no < 6; no++)
{
      cout << message5[no];
}
cout << " ";

for (no = 4; no >= 0; no--)
{
      temp3 = read3 % 100;
      read3 = read3 / 100;
      message10[no] = (temp3 + 55);
}
for (no = 0; no < 5; no++)
{
      message9[no] = char(message10[no]);
}
for (no = 0; no < 5; no++)
{
      cout << message9[no];
}
cout << " ";

for (no = 6; no >= 0; no--)
{
      temp4 = read4 % 100;
```

```
                              read4 = read4 / 100;
                              message12[no] = (temp4 + 55);
                         }
                         for (no = 0; no < 8; no++)
                         {
                              message11[no] = char(message12[no]);
                         }
                         for (no = 0; no < 8; no++)
                         {
                              cout << message11[no];
                         }
                         cout << " ";

                         for (no = 5; no >= 0; no--)
                         {
                              temp5 = read5 % 100;
                              read5 = read5 / 100;
                              message14[no] = (temp5 + 55);
                         }
                         for (no = 0; no < 7; no++)
                         {
                              message13[no] = char(message14[no]);
                         }
                         for (no = 0; no < 7; no++)
                         {
                              cout << message13[no];
                         }
                         cout << endl;

                         cout << "Client message 3: CCerts KeyEx
CertVerif MastSecr" << endl;
                         write0 = 124446596160;
                         write2 = 2046661465;
                         write3 = 124659613146595047;
                         write4 = 2242606128464459;

                         b.write(write0);
                         c.write(write2);
                         h.write(write3);
                         i.write(write4);

                         wait(2);

                         sc_uint<64> f, g, write8, write9, write10,
read6, read7, read8, read9;
                         temp = temp2 = 0;
                         g = a.read();
                         read6 = d.read();
                         if(g == 2242606128464459 && read6 ==
2846606029504452)
                         {
                              for (no = 7; no >= 0; no--)
```

```
                                  {
                                        temp = g % 100;
                                        g = g / 100;
                                        message16[no] = (temp + 55);
                                  }
                                  for (no = 0; no < 8; no++)
                                  {
                                        message15[no] = 
char(message16[no]);
                                  }
                                  cout << "Client receives: ";
                                  for (no = 0; no < 8; no++)
                                  {
                                        cout << message15[no];
                                  }

                                  for (no = 7; no >= 0; no--)
                                  {
                                        temp2 = read6 % 100;
                                        read6 = read6 / 100;
                                        message18[no] = (temp2 + 55);
                                  }
                                  for (no = 0; no < 8; no++)
                                  {
                                        message17[no] = 
char(message18[no]);
                                  }
                                  cout << " ";
                                  for (no = 0; no < 8; no++)
                                  {
                                        cout << message17[no];
                                  }
                                  cout << endl;

                                  cout << "Client message 4: 
ChangeCipherSpec Finished" << endl;
                                  write8 = 1249425548461250;
                                  write9 = 5749465928574644;
                                  write10 = 1550555060494645;
                                  b.write(write8);
                                  c.write(write9);
                                  h.write(write10);

                                  wait(2);

                                  temp = temp2 = temp3 = 0;
                                  read7 = a.read();
                                  read8 = d.read();
                                  read9 = e.read();

                                  if(read7 == 1249425548461250 && read8 
== 5749465928574644 && read9 == 1550555060494645)
```

```
                                        {
                                             for (no = 7; no >= 0; no--)
                                             {
                                                  temp = read7 % 100;
                                                  read7 = read7 / 100;
                                                  message20[no] = (temp +
55);
                                             }
                                             for (no = 0; no < 8; no++)
                                             {
                                                  message19[no] =
char(message20[no]);
                                             }
                                             cout << "Client receives: ";
                                             for (no = 0; no < 8; no++)
                                             {
                                                  cout << message19[no];
                                             }

                                             for (no = 7; no >= 0; no--)
                                             {
                                                  temp2 = read8 % 100;
                                                  read8 = read8 / 100;
                                                  message22[no] = (temp2 +
55);
                                             }
                                             for (no = 0; no < 8; no++)
                                             {
                                                  message21[no] =
char(message22[no]);
                                             }
                                             for (no = 0; no < 8; no++)
                                             {
                                                  cout << message21[no];
                                             }
                                             cout << " ";

                                             for (no = 7; no >= 0; no--)
                                             {
                                                  temp3 = read9 % 100;
                                                  read9 = read9 / 100;
                                                  message24[no] = (temp3 +
55);
                                             }
                                             for (no = 0; no < 8; no++)
                                             {
                                                  message23[no] =
char(message24[no]);
                                             }
                                             for (no = 0; no < 8; no++)
                                             {
                                                  cout << message23[no];
                                             }
                                             cout << endl << endl;

                                             wait(3);

                                             cout << "Enduser to Sensor1:
Client Hello" << endl;
                                             write1 = 125350465561;
                                             write2 = 1746535356;
                                             m.write(write1);
                                             n.write(write2);

                                             wait(2);

                                             read1 = o.read();
                                             read2 = p.read();
                                             read3 = q.read();
                                             read4 = r.read();
                                             read5 = s.read();
                                             read6 = t.read();
                                             read7 = u.read();
                                             if(read1 == 284659634659 &&
read2 == 1746535356 && read3 == 2846606029504452 && read4 ==
2242606128464459 && read5 == 1249425548461250 && read6 ==
5749465928574644 && read7 == 1550555060494645)
                                             {
                                                  for (no = 5; no >= 0; no--)
                                                  {
                                                       temp = read1 % 100;
                                                       read1 = read1 / 100;
                                                       message16[no] = (temp
+ 55);
                                                  }
                                                  for (no = 0; no < 6; no++)
                                                  {
                                                       message15[no] =
char(message16[no]);
                                                  }
                                                  cout << "Client receives:
";
                                                  for (no = 0; no < 6; no++)
                                                  {
                                                       cout <<
message15[no];
                                                  }
                                                  cout << " ";

                                                  for (no = 4; no >= 0; no--)
                                                  {
                                                       temp = read2 % 100;
                                                       read2 = read2 / 100;
```

```
                                                 message16[no] = (temp
+ 55);
                                            }
                                            for (no = 0; no < 5; no++)
                                            {
                                                 message15[no] =
char(message16[no]);
                                            }
                                            for (no = 0; no < 5; no++)
                                            {
                                                 cout <<
message15[no];
                                            }
                                            cout << " ";

                                            for (no = 7; no >= 0; no--)
                                            {
                                                 temp = read3 % 100;
                                                 read3 = read3 / 100;
                                                 message16[no] = (temp
+ 55);
                                            }
                                            for (no = 0; no < 8; no++)
                                            {
                                                 message15[no] =
char(message16[no]);
                                            }
                                            for (no = 0; no < 8; no++)
                                            {
                                                 cout <<
message15[no];
                                            }
                                            cout << " ";

                                            for (no = 7; no >= 0; no--)
                                            {
                                                 temp = read4 % 100;
                                                 read4 = read4 / 100;
                                                 message16[no] = (temp
+ 55);
                                            }
                                            for (no = 0; no < 8; no++)
                                            {
                                                 message15[no] =
char(message16[no]);
                                            }
                                            for (no = 0; no < 8; no++)
                                            {
                                                 cout <<
message15[no];
                                            }
                                            cout << " ";
```

```
                                        for (no = 7; no >= 0; no--)
                                        {
                                             temp = read5 % 100;
                                             read5 = read5 / 100;
                                             message16[no] = (temp
+ 55);
                                        }
                                        for (no = 0; no < 8; no++)
                                        {
                                             message15[no] =
char(message16[no]);
                                        }
                                        for (no = 0; no < 8; no++)
                                        {
                                             cout <<
message15[no];
                                        }

                                        for (no = 7; no >= 0; no--)
                                        {
                                             temp = read6 % 100;
                                             read6 = read6 / 100;
                                             message16[no] = (temp
+ 55);
                                        }
                                        for (no = 0; no < 8; no++)
                                        {
                                             message15[no] =
char(message16[no]);
                                        }
                                        for (no = 0; no < 8; no++)
                                        {
                                             cout <<
message15[no];
                                        }
                                        cout << " ";

                                        for (no = 7; no >= 0; no--)
                                        {
                                             temp = read7 % 100;
                                             read7 = read7 / 100;
                                             message16[no] = (temp
+ 55);
                                        }
                                        for (no = 0; no < 8; no++)
                                        {
                                             message15[no] =
char(message16[no]);
                                        }
                                        for (no = 0; no < 8; no++)
                                        {
```

```
                                                        cout <<
message15[no];
                                                  }
                                                  cout << endl;

                                                  write1 = 1249425548461250;
                                                  write2 = 5749465928574644;
                                                  write3 = 1550555060494645;
                                                  write4 = 28464462594645;
                                                  m.write(write1);
                                                  n.write(write2);
                                                  v.write(write3);
                                                  w.write(write4);
                                                  cout << "Enduser to
Sensor1: ChangeCipherSpec Finished Secured" << endl;
                                                  wait(4);

                                                  cout << "Enduser to
Sensor2: Client Hello" << endl;
                                                  write1 = 125350465561;
                                                  write2 = 1746535356;
                                                  x.write(write1);
                                                  y.write(write2);
                                                  wait(2);

                                                  read1 = z.read();
                                                  read2 = aa.read();
                                                  read3 = ab.read();
                                                  read4 = ac.read();
                                                  read5 = ad.read();
                                                  read6 = ae.read();
                                                  read7 = af.read();
                                                  if(read1 == 284659634659 &&
read2 == 1746535356 && read3 == 2846606029504452 && read4 ==
2242606128464459 && read5 == 1249425548461250 && read6 ==
5749465928574644 && read7 == 1550555060494645)
                                                  {
                                                        for (no = 5; no >= 0;
no--)
                                                        {
                                                              temp = read1 %
100;
                                                              read1 = read1 /
100;
                                                              message16[no] =
(temp + 55);
                                                        }
                                                        for (no = 0; no < 6;
no++)
                                                        {
                                                              message15[no] =
char(message16[no]);
}
cout << "Client receives: ";
for (no = 0; no < 6; no++)
{
     cout << message15[no];
}
cout << " ";
for (no = 4; no >= 0; no--)
{
     temp = read2 % 100;
     read2 = read2 / 100;
     message16[no] = (temp + 55);
}
for (no = 0; no < 5; no++)
{
     message15[no] = char(message16[no]);
}
for (no = 0; no < 5; no++)
{
     cout << message15[no];
}
cout << " ";
for (no = 7; no >= 0; no--)
{
     temp = read3 % 100;
     read3 = read3 / 100;
     message16[no] = (temp + 55);
}
for (no = 0; no < 8; no++)
{
     message15[no] = char(message16[no]);
}
```

```
                                        for (no = 0; no < 8;
no++)
                                        {
                                             cout <<
message15[no];
                                        }
                                        cout << " ";

                                        for (no = 7; no >= 0;
no--)
                                        {
                                             temp = read4 %
100;
                                             read4 = read4 /
100;
                                             message16[no] =
(temp + 55);
                                        }
                                        for (no = 0; no < 8;
no++)
                                        {
                                             message15[no] =
char(message16[no]);
                                        }
                                        for (no = 0; no < 8;
no++)
                                        {
                                             cout <<
message15[no];
                                        }
                                        cout << " ";

                                        for (no = 7; no >= 0;
no--)
                                        {
                                             temp = read5 %
100;
                                             read5 = read5 /
100;
                                             message16[no] =
(temp + 55);
                                        }
                                        for (no = 0; no < 8;
no++)
                                        {
                                             message15[no] =
char(message16[no]);
                                        }
                                        for (no = 0; no < 8;
no++)
                                        {
```

```
     cout << message15[no];
}
for (no = 7; no >= 0; no--)
{
     temp = read6 % 100;
     read6 = read6 / 100;
     message16[no] = (temp + 55);
}
for (no = 0; no < 8; no++)
{
     message15[no] = char(message16[no]);
}
for (no = 0; no < 8; no++)
{
     cout << message15[no];
}
cout << " ";
for (no = 7; no >= 0; no--)
{
     temp = read7 % 100;
     read7 = read7 / 100;
     message16[no] = (temp + 55);
}
for (no = 0; no < 8; no++)
{
     message15[no] = char(message16[no]);
}
for (no = 0; no < 8; no++)
{
     cout << message15[no];
}
cout << endl;
```

```
                                                write1 =
1249425548461250;
                                                write2 =
5749465928574644;
                                                write3 =
1550555060494645;
                                                write4 =
28464462594645;
                                                x.write(write1);
                                                y.write(write2);
                                                ag.write(write3);
                                                ah.write(write4);
                                                cout << "Enduser to
Sensor2: ChangeCipherSpec Finished Secured" << endl;
                                                wait(4);

                                                cout << "Enduser to
Sensor3: Client Hello" << endl;
                                                write1 =
125350465561;
                                                write2 = 1746535356;
                                                ai.write(write1);
                                                aj.write(write2);
                                                wait(2);

                                                read1 = ak.read();
                                                read2 = al.read();
                                                read3 = am.read();
                                                read4 = an.read();
                                                read5 = ao.read();
                                                read6 = ap.read();
                                                read7 = aq.read();
                                                if(read1 ==
284659634659 && read2 == 1746535356 && read3 == 2846606029504452 &&
read4 == 2242606128464459 && read5 == 1249425548461250 && read6 ==
5749465928574644 && read7 == 1550555060494645)
                                                {
                                                     for (no = 5; no
>= 0; no--)
                                                     {
                                                          temp =
read1 % 100;
                                                          read1 =
read1 / 100;

     message16[no] = (temp + 55);
                                                     }
                                                     for (no = 0; no
< 6; no++)
                                                     {

     message15[no] = char(message16[no]);
```

```
                                                      }
                                                      cout << "Client
receives: ";
                                                      for (no = 0; no
< 6; no++)
                                                      {
                                                           cout <<
message15[no];
                                                      }
                                                      cout << " ";

                                                      for (no = 4; no
>= 0; no--)
                                                      {
                                                           temp =
read2 % 100;
                                                           read2 =
read2 / 100;

     message16[no] = (temp + 55);
                                                      }
                                                      for (no = 0; no
< 5; no++)
                                                      {

     message15[no] = char(message16[no]);
                                                      }
                                                      for (no = 0; no
< 5; no++)
                                                      {
                                                           cout <<
message15[no];
                                                      }
                                                      cout << " ";

                                                      for (no = 7; no
>= 0; no--)
                                                      {
                                                           temp =
read3 % 100;
                                                           read3 =
read3 / 100;

     message16[no] = (temp + 55);
                                                      }
                                                      for (no = 0; no
< 8; no++)
                                                      {

     message15[no] = char(message16[no]);
                                                      }
```

```
                                                       for (no = 0; no
< 8; no++)
                                                       {
                                                            cout <<
message15[no];
                                                       }
                                                       cout << " ";

                                                       for (no = 7; no
>= 0; no--)
                                                       {
                                                            temp =
read4 % 100;
                                                            read4 =
read4 / 100;
     message16[no] = (temp + 55);
                                                       }
                                                       for (no = 0; no
< 8; no++)
                                                       {

     message15[no] = char(message16[no]);
                                                       }
                                                       for (no = 0; no
< 8; no++)
                                                       {
                                                            cout <<
message15[no];
                                                       }
                                                       cout << " ";

                                                       for (no = 7; no
>= 0; no--)
                                                       {
                                                            temp =
read5 % 100;
                                                            read5 =
read5 / 100;
     message16[no] = (temp + 55);
                                                       }
                                                       for (no = 0; no
< 8; no++)
                                                       {

     message15[no] = char(message16[no]);
                                                       }
                                                       for (no = 0; no
< 8; no++)
                                                       {
```

```
                                                                      cout <<
message15[no];
                                                                 }
                                                                 for (no = 7; no
>= 0; no--)
                                                                 {
                                                                      temp =
read6 % 100;
                                                                      read6 =
read6 / 100;

     message16[no] = (temp + 55);
                                                                 }
                                                                 for (no = 0; no
< 8; no++)
                                                                 {

     message15[no] = char(message16[no]);
                                                                 }
                                                                 for (no = 0; no
< 8; no++)
                                                                 {
                                                                      cout <<
message15[no];
                                                                 }
                                                                 cout << " ";

                                                                 for (no = 7; no
>= 0; no--)
                                                                 {
                                                                      temp =
read7 % 100;
                                                                      read7 =
read7 / 100;

     message16[no] = (temp + 55);
                                                                 }
                                                                 for (no = 0; no
< 8; no++)
                                                                 {

     message15[no] = char(message16[no]);
                                                                 }
                                                                 for (no = 0; no
< 8; no++)
                                                                 {
                                                                      cout <<
message15[no];
                                                                 }
                                                                 cout << endl;
```

```
                                                       write1 =
1249425548461250;
                                                       write2 =
5749465928574644;
                                                       write3 =
1550555060494645;
                                                       write4 =
28464462594645;

     ai.write(write1);

     aj.write(write2);

     ar.write(write3);

     as.write(write4);
                                                       cout << "Enduser
to Sensor3: ChangeCipherSpec Finished Secured" << endl;
                                                       wait(4);

                                                       cout << "Enduser
to Sensor4: Client Hello" << endl;
                                                       write1 =
125350465561;
                                                       write2 =
1746535356;

     at.write(write1);

     au.write(write2);
                                                       wait(2);

                                                       read1 =
av.read();
                                                       read2 =
aw.read();
                                                       read3 =
ax.read();
                                                       read4 =
ay.read();
                                                       read5 =
az.read();
                                                       read6 =
ba.read();
                                                       read7 =
bb.read();
                                                       if(read1 ==
284659634659 && read2 == 1746535356 && read3 == 2846606029504452 &&
read4 == 2242606128464459 && read5 == 1249425548461250 && read6 ==
5749465928574644 && read7 == 1550555060494645)
                                                       {
                                                            for (no =
5; no >= 0; no--)
                                                            {
                                                                 temp
= read1 % 100;
                                                                 read1
= read1 / 100;

     message16[no] = (temp + 55);
                                                            }
                                                            for (no =
0; no < 6; no++)
                                                            {

     message15[no] = char(message16[no]);
                                                            }
                                                            cout <<
"Client receives: ";
                                                            for (no =
0; no < 6; no++)
                                                            {
                                                                 cout
<< message15[no];
                                                            }
                                                            cout << "
";

                                                            for (no =
4; no >= 0; no--)
                                                            {
                                                                 temp
= read2 % 100;
                                                                 read2
= read2 / 100;
     message16[no] = (temp + 55);
                                                            }
                                                            for (no =
0; no < 5; no++)
                                                            {

     message15[no] = char(message16[no]);
                                                            }
                                                            for (no =
0; no < 5; no++)
                                                            {
                                                                 cout
<< message15[no];
                                                            }
                                                            cout << "
";
```

```
                                                            for (no =
7; no >= 0; no--)
                                                            {
                                                                 temp
= read3 % 100;
                                                                 read3
= read3 / 100;
     message16[no] = (temp + 55);
                                                            }
                                                            for (no =
0; no < 8; no++)
                                                            {
     message15[no] = char(message16[no]);
                                                            }
                                                            for (no =
0; no < 8; no++)
                                                            {
                                                                 cout
<< message15[no];
                                                            }
                                                            cout << "
";
                                                            for (no =
7; no >= 0; no--)
                                                            {
                                                                 temp
= read4 % 100;
                                                                 read4
= read4 / 100;
     message16[no] = (temp + 55);
                                                            }
                                                            for (no =
0; no < 8; no++)
                                                            {
     message15[no] = char(message16[no]);
                                                            }
                                                            for (no =
0; no < 8; no++)
                                                            {
                                                                 cout
<< message15[no];
                                                            }
                                                            cout << "
";
                                                            for (no =
7; no >= 0; no--)
```

```
                                                    {
                                                         temp
= read5 % 100;
                                                         read5
= read5 / 100;
     message16[no] = (temp + 55);
                                                    }
                                                    for (no =
0; no < 8; no++)
                                                    {
     message15[no] = char(message16[no]);
                                                    }
                                                    for (no =
0; no < 8; no++)
                                                    {
                                                         cout
<< message15[no];
                                                    }

                                                    for (no =
7; no >= 0; no--)
                                                    {
                                                         temp
= read6 % 100;
                                                         read6
= read6 / 100;
     message16[no] = (temp + 55);
                                                    }
                                                    for (no =
0; no < 8; no++)
                                                    {
     message15[no] = char(message16[no]);
                                                    }
                                                    for (no =
0; no < 8; no++)
                                                    {
                                                         cout
<< message15[no];
                                                    }
                                                    cout << "
";

                                                    for (no =
7; no >= 0; no--)
                                                    {
                                                         temp
= read7 % 100;
```

```
                                                                                read7
= read7 / 100;

     message16[no] = (temp + 55);
                                                                           }
                                                                           for (no =
0; no < 8; no++)
                                                                           {

     message15[no] = char(message16[no]);
                                                                           }
                                                                           for (no =
0; no < 8; no++)
                                                                           {
                                                                                cout
<< message15[no];
                                                                           }
                                                                           cout <<
endl;

                                                                           write1 =
1249425548461250;
                                                                           write2 =
5749465928574644;
                                                                           write3 =
1550555060494645;
                                                                           write4 =
28464462594645;

     at.write(write1);

     au.write(write2);

     bc.write(write3);

     bd.write(write4);
                                                                           cout <<
"Enduser to Sensor4: ChangeCipherSpec Finished Secured" << endl;
                                                                           wait(4);

                                                                           cout <<
"Enduser to Sensor5: Client Hello" << endl;
                                                                           write1 =
125350465561;
                                                                           write2 =
1746535356;

     be.write(write1);

     bf.write(write2);
                                                                           wait(2);
```

```
                                                           read1 =
bg.read();
                                                           read2 =
bh.read();
                                                           read3 =
bi.read();
                                                           read4 =
bj.read();
                                                           read5 =
bk.read();
                                                           read6 =
bl.read();
                                                           read7 =
bm.read();
                                                           if(read1
== 284659634659 && read2 == 1746535356 && read3 == 2846606029504452 &&
read4 == 2242606128464459 && read5 == 1249425548461250 && read6 ==
5749465928574644 && read7 == 1550555060494645)
                                                           {
                                                                for
(no = 5; no >= 0; no--)
                                                                {

     temp = read1 % 100;

     read1 = read1 / 100;

     message16[no] = (temp + 55);
                                                                }
                                                                for
(no = 0; no < 6; no++)
                                                                {

     message15[no] = char(message16[no]);
                                                                }
                                                                cout
<< "Client receives: ";
                                                                for
(no = 0; no < 6; no++)
                                                                {

     cout << message15[no];
                                                                }
                                                                cout
<< " ";

                                                                for
(no = 4; no >= 0; no--)
                                                                {

     temp = read2 % 100;
```

```
          read2 = read2 / 100;

          message16[no] = (temp + 55);
                                                                             }
                                                                             for
     (no = 0; no < 5; no++)
                                                                             {

          message15[no] = char(message16[no]);
                                                                             }
                                                                             for
     (no = 0; no < 5; no++)
                                                                             {

          cout << message15[no];
                                                                             }
                                                                             cout
     << " ";

                                                                             for
     (no = 7; no >= 0; no--)
                                                                             {

          temp = read3 % 100;

          read3 = read3 / 100;

          message16[no] = (temp + 55);
                                                                             }
                                                                             for
     (no = 0; no < 8; no++)
                                                                             {

          message15[no] = char(message16[no]);
                                                                             }
                                                                             for
     (no = 0; no < 8; no++)
                                                                             {

          cout << message15[no];
                                                                             }
                                                                             cout
     << " ";

                                                                             for
     (no = 7; no >= 0; no--)
                                                                             {

          temp = read4 % 100;

          read4 = read4 / 100;
     message16[no] = (temp + 55);
                                                                  }
                                                                  for
(no = 0; no < 8; no++)
                                                                  {

     message15[no] = char(message16[no]);
                                                                  }
                                                                  for
(no = 0; no < 8; no++)
                                                                  {

     cout << message15[no];
                                                                  }
                                                                  cout
<< " ";

                                                                  for
(no = 7; no >= 0; no--)
                                                                  {

     temp = read5 % 100;

     read5 = read5 / 100;

     message16[no] = (temp + 55);
                                                                  }
                                                                  for
(no = 0; no < 8; no++)
                                                                  {

     message15[no] = char(message16[no]);
                                                                  }
                                                                  for
(no = 0; no < 8; no++)
                                                                  {

     cout << message15[no];
                                                                  }

                                                                  for
(no = 7; no >= 0; no--)
                                                                  {

     temp = read6 % 100;

     read6 = read6 / 100;

     message16[no] = (temp + 55);
                                                                  }
                                                                  for
(no = 0; no < 8; no++)
                                                                  {
     message15[no] = char(message16[no]);
                                                                  }
                                                                  for
(no = 0; no < 8; no++)
                                                                  {
     cout << message15[no];
                                                                  }
                                                                  cout
<< " ";

                                                                  for
(no = 7; no >= 0; no--)
                                                                  {
     temp = read7 % 100;

     read7 = read7 / 100;

     message16[no] = (temp + 55);
                                                                  }
                                                                  for
(no = 0; no < 8; no++)
                                                                  {
     message15[no] = char(message16[no]);
                                                                  }
                                                                  for
(no = 0; no < 8; no++)
                                                                  {
     cout << message15[no];
                                                                  }
                                                                  cout
<< endl;

     write1 = 1249425548461250;

     write2 = 5749465928574644;

     write3 = 1550555060494645;

     write4 = 28464462594645;

     be.write(write1);

     bf.write(write2);
```

```
      bn.write(write3);

      bo.write(write4);
                                                                   cout
<< "Enduser to Sensor5: ChangeCipherSpec Finished Secured" << endl;

      wait(4);

                                                                   for
(temp = 0; temp < 6; temp++)
                                                                   {

      read0 = a.read();

      cout << "From Sensor1: " << read0 << endl;

      wait(3);


      read0 = a.read();

      cout << "From Sensor2: " << read0 << endl;

      wait(3);


      read0 = a.read();

      cout << "From Sensor3: " << read0 << endl;

      wait(3);


      read0 = a.read();

      cout << "From Sensor4: " << read0 << endl;

      wait(3);


      read0 = a.read();

      cout << "From Sensor5: " << read0 << endl;

      wait(3);
                                                                   }
                                                              }
                                                        }
                                                  }
                                            }
```

```
                                    }
                              }

                        }

                  }

            }

      }

      sc_stop();
  }
}
```

Node2.h

```
SC_MODULE(node_2)
{
      sc_in_clk clk;
      sc_in<sc_uint<64>> a;
      sc_out<sc_uint<64>> b;
      sc_in<sc_uint<64>> c;
      sc_out<sc_uint<64>> d;
      sc_out<sc_uint<64>> e;
      sc_out<sc_uint<64>> f;
      sc_out<sc_uint<64>> g;
      sc_in<sc_uint<64>> h;
      sc_in<sc_uint<64>> i;
      sc_in<sc_uint<64>> j;
      sc_in<sc_uint<64>> k;
      sc_in<sc_uint<64>> l;
      sc_out<sc_uint<64>> n;
      sc_out<sc_uint<64>> o;
      sc_out<sc_uint<64>> p;
      sc_out<sc_uint<64>> q;
      sc_out<sc_uint<64>> r;
      sc_out<sc_uint<64>> s;
      sc_out<sc_uint<64>> t;
      sc_out<sc_uint<64>> u;
      sc_out<sc_uint<64>> v;
      sc_out<sc_uint<64>> w;
      sc_out<sc_uint<64>> x;
      sc_out<sc_uint<64>> y;
      sc_out<sc_uint<64>> z;
      sc_out<sc_uint<64>> aa;
      sc_out<sc_uint<64>> ab;
      sc_in<sc_uint<64>> ac;
      sc_in<sc_uint<64>> ad;
```

```
	sc_in<sc_uint<64>> ae;
	sc_in<sc_uint<64>> af;
	sc_in<sc_uint<64>> ag;


	SC_CTOR(node_2)
	{
		SC_CTHREAD(connect2, clk.pos());
	}

	void connect2();
};
```

Node2.cpp

```
#include <systemc.h>
#include "Node2.h"

void node_2::connect2()
{
	wait();
	while(true)
	{
		sc_uint<64> write0;
		sc_uint<64> read0;
		sc_uint<64> read1;
		sc_uint<64> write1;
		read0 = a.read();
		read1 = c.read();
		int temp, temp2, temp3, temp4, temp5, temp6, temp7;
		int no;
		int message1[7], message3[6], message6[7], message8[6],
message10[7], message12[10], message14[9], message16[9], message18[9],
message20[9];
		char message2[7], message4[6], message5[7], message7[6],
message9[7], message11[10], message13[9], message15[9], message17[9],
message19[9];

		if (read0 == 1746535356 && read1 == 254259425460)
		{
			for (no = 4; no >= 0; no--)
			{
				temp = read0 % 100;
				read0 = read0 / 100;
				message3[no] = (temp + 55);
			}
			for (no = 0; no < 5; no++)
			{
				message4[no] = char(message3[no]);
			}
```

```
cout << "Server receives: ";
for (no = 0; no < 5; no++)
{
      cout << message4[no];
}
cout << " " ;
for (no = 5; no >= 0; no--)
{
      temp2 = read1 % 100;
      read1 = read1 / 100;
      message1[no] = (temp2 + 55);
}
for (no = 0; no < 6; no++)
{
      message2[no] = char(message1[no]);
}
for (no = 0; no < 6; no++)
{
      cout << message2[no];
}
cout << endl;

cout << "Server message 1: Hello Params" << endl;
write0 = 1746535356;
write1 = 254259425460;
b.write(write0);
d.write(write1);

wait(2);

sc_uint<64> write2, write3, write4, write5;
temp = 0;
read0 = a.read();
if(read0 == 125656525046)
{
      for (no = 5; no >= 0; no--)
      {
            temp = read0 % 100;
            read0 = read0 / 100;
            message6[no] = (temp + 55);
      }

      for (no = 0; no < 6; no++)
      {
            message5[no] = char(message6[no]);
      }

      cout << "Server receives: ";

      for (no = 0; no < 6; no++)
      {
            cout << message5[no];
```

```
                    }

                    cout << endl;

                    cout << "Server message 2: Hello SCerts KeyEx
CertReq Finish" << endl;
                    write1 = 1746535356;
                    write2 = 281246596160;
                    write3 = 2046661465;
                    write4 = 12465961274658;
                    write5 = 155055506049;
                    b.write(write1);
                    d.write(write2);
                    e.write(write3);
                    f.write(write4);
                    g.write(write5);

                    wait(2);

                    sc_uint<64> f, read2, read3, read4, read5, read6,
read7, write6, write7, write8, write9, write10;
                    sc_uint<64> g;
                    temp = temp2 = temp3 = temp4 = temp5 = temp6 =
temp7 = 0;
                    f = a.read();
                    read2 = c.read();
                    read3 = h.read();
                    read4 = i.read();

                    if(f == 124446596160 && read2 == 2046661465 &&
read3 == 124659613146595047 && read4 == 2242606128464459)
                    {
                         for (no = 5; no >= 0; no--)
                         {
                              temp = f % 100;
                              f = f / 100;
                              message10[no] = (temp + 55);
                         }
                         for (no = 0; no < 6; no++)
                         {
                              message9[no] = char(message10[no]);
                         }
                         cout << "Server receives: ";
                         for (no = 0; no < 6; no++)
                         {
                              cout << message9[no];
                         }
                         cout << " ";

                         for (no = 4; no >= 0; no--)
                         {
                              temp2 = read2 % 100;
```

```
		read2 = read2 / 100;
		message8[no] = (temp2 + 55);
	}
	for (no = 0; no < 5; no++)
	{
		message7[no] = char(message8[no]);
	}
	for (no = 0; no < 5; no++)
	{
		cout << message7[no];
	}
	cout << " ";

	for (no = 8; no >= 0; no--)
	{
		temp3 = read3 % 100;
		read3 = read3 / 100;
		message12[no] = (temp3 + 55);
	}
	for (no = 0; no < 9; no++)
	{
		message11[no] = char(message12[no]);
	}
	for (no = 0; no < 9; no++)
	{
		cout << message11[no];
	}
	cout << " ";

	for (no = 7; no >= 0; no--)
	{
		temp4 = read4 % 100;
		read4 = read4 / 100;
		message14[no] = (temp4 + 55);
	}
	for (no = 0; no < 8; no++)
	{
		message13[no] = char(message14[no]);
	}
	for (no = 0; no < 8; no++)
	{
		cout << message13[no];
	}
	cout << " ";

	cout << endl;

	cout << "Server message 3: MastSecr
SessTick" << endl;
	write6 = 2242606128464459;
	write7 = 2846606029504452;
	b.write(write6);
```

```
                        d.write(write7);

                        wait(2);

                        read5 = a.read();
                        read6 = c.read();
                        read7 = h.read();

                        if(read5 == 1249425548461250 && read6 == 
5749465928574644 && read7 == 1550555060494645)
                        {
                              for (no = 7; no >= 0; no--)
                              {
                                    temp5 = read5 % 100;
                                    read5 = read5 / 100;
                                    message16[no] = (temp5 + 55);
                              }
                              for (no = 0; no < 8; no++)
                              {
                                    message15[no] = 
char(message16[no]);
                              }
                              cout << "Server receives: ";
                              for (no = 0; no < 8; no++)
                              {
                                    cout << message15[no];
                              }

                              for (no = 7; no >= 0; no--)
                              {
                                    temp6 = read6 % 100;
                                    read6 = read6 / 100;
                                    message18[no] = (temp6 + 55);
                              }
                              for (no = 0; no < 8; no++)
                              {
                                    message17[no] = 
char(message18[no]);
                              }
                              for (no = 0; no < 8; no++)
                              {
                                    cout << message17[no];
                              }
                              cout << " ";

                              for (no = 7; no >= 0; no--)
                              {
                                    temp7 = read7 % 100;
                                    read7 = read7 / 100;
                                    message20[no] = (temp7 + 55);
                              }
                              for (no = 0; no < 8; no++)
```

```
                                {
                                     message19[no] = 
char(message20[no]);
                                }
                                for (no = 0; no < 8; no++)
                                {
                                     cout << message19[no];
                                }
                                cout << " " << endl;

                                write8 = 1249425548461250;
                                write9 = 5749465928574644;
                                write10 = 1550555060494645;
                                b.write(write8);
                                d.write(write9);
                                e.write(write10);
                                cout << "Server message 4: 
ChangeCipherSpec Finished" << endl;

                                wait(2);

                                cout << "Gateway to Sensor1: Enduser 
Authorized" << endl;
                                sc_uint<64> write11, write12, write13;
                                write11 = 14554562604659;
                                write12 = 1062614956;
                                write13 = 5950674645;
                                n.write(write11);
                                o.write(write12);
                                p.write(write13);
                                wait(6);

                                cout << "Gateway to Sensor2: Enduser 
Authorized" << endl;
                                write11 = 14554562604659;
                                write12 = 1062614956;
                                write13 = 5950674645;
                                q.write(write11);
                                r.write(write12);
                                s.write(write13);
                                wait(6);

                                cout << "Gateway to Sensor3: Enduser 
Authorized" << endl;
                                write11 = 14554562604659;
                                write12 = 1062614956;
                                write13 = 5950674645;
                                t.write(write11);
                                u.write(write12);
                                v.write(write13);
                                wait(6);
```

```
                                        cout << "Gateway to Sensor4: Enduser
Authorized" << endl;
                                        write11 = 14554562604659;
                                        write12 = 1062614956;
                                        write13 = 5950674645;
                                        w.write(write11);
                                        x.write(write12);
                                        y.write(write13);
                                        wait(6);

                                        cout << "Gateway to Sensor5: Enduser
Authorized" << endl;
                                        write11 = 14554562604659;
                                        write12 = 1062614956;
                                        write13 = 5950674645;
                                        z.write(write11);
                                        aa.write(write12);
                                        ab.write(write13);
                                        wait(7);

                                        for (temp = 0; temp < 6; temp++)
                                        {
                                              read0 = ac.read();
                                              b.write(read0);
                                              wait(3);

                                              read0 = ad.read();
                                              b.write(read0);
                                              wait(3);

                                              read0 = ae.read();
                                              b.write(read0);
                                              wait(3);

                                              read0 = af.read();
                                              b.write(read0);
                                              wait(3);

                                              read0 = ag.read();
                                              b.write(read0);
                                              wait(3);
                                        }
                                  }

                            }

                      }
                }

                sc_stop();
          }
    }
```

Sensor1.h

```
SC_MODULE(sensor_1)
{
      sc_in_clk clk;
      sc_in<sc_uint<64>> a;
      sc_in<sc_uint<64>> b;
      sc_in<sc_uint<64>> c;
      sc_in<sc_uint<64>> d;
      sc_in<sc_uint<64>> e;
      sc_out<sc_uint<64>> f;
      sc_out<sc_uint<64>> g;
      sc_out<sc_uint<64>> h;
      sc_out<sc_uint<64>> i;
      sc_out<sc_uint<64>> j;
      sc_out<sc_uint<64>> k;
      sc_out<sc_uint<64>> l;
      sc_in<sc_uint<64>> m;
      sc_in<sc_uint<64>> n;
      sc_out<sc_uint<64>> o;

      SC_CTOR(sensor_1)
      {
            SC_CTHREAD(connect3, clk.pos());
      }

      void connect3();
};
```

Sensor1.cpp

```
#include <systemc.h>
#include "sensor1.h"
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>

using namespace std;

void sensor_1::connect3()
{
      wait(10);
      while(true)
      {
            sc_uint<64> read1, read2, read3, read4, write1, write2,
write3, write4, write5, write6, write7;
            int no, temp, temp2, temp3;
            read1 = a.read();
            read2 = b.read();
            read3 = c.read();
            char message1[9], message3[6], message5[6];
            int message2[9], message4[6], message6[6];

            if(read1 == 14554562604659 && read2 == 1062614956 && read3
== 5950674645)
            {
                  for (no = 6; no >= 0; no--)
                  {
                        temp = read1 % 100;
                        read1 = read1 / 100;
                        message2[no] = (temp + 55);
                  }
                  for (no = 0; no < 7; no++)
                  {
                        message1[no] = char(message2[no]);
                  }
                  cout << "Sensor1 receives: ";
                  for (no = 0; no < 7; no++)
                  {
                        cout << message1[no];
                  }
                  cout << " ";

                  for (no = 4; no >= 0; no--)
                  {
                        temp2 = read2 % 100;
                        read2 = read2 / 100;
                        message4[no] = (temp2 + 55);
                  }
                  for (no = 0; no < 5; no++)
                  {
                        message3[no] = char(message4[no]);
                  }
                  for (no = 0; no < 5; no++)
                  {
                        cout << message3[no];
                  }

                  for (no = 4; no >= 0; no--)
                  {
                        temp3 = read3 % 100;
                        read3 = read3 / 100;
                        message6[no] = (temp3 + 55);
                  }
                  for (no = 0; no < 5; no++)
                  {
                        message5[no] = char(message6[no]);
                  }
                  for (no = 0; no < 5; no++)
                  {
                        cout << message5[no];
                  }
                  cout << endl;
```

```
wait(2);

read1 = d.read();
read2 = e.read();
if(read1 == 125350465561 && read2 == 1746535356)
{
      for (no = 5; no >= 0; no--)
      {
            temp = read1 % 100;
            read1 = read1 / 100;
            message2[no] = (temp + 55);
      }
      for (no = 0; no < 6; no++)
      {
            message1[no] = char(message2[no]);
      }
      cout << "Sensor1 receives: ";
      for (no = 0; no < 6; no++)
      {
            cout << message1[no];
      }
      cout << " ";

      for (no = 4; no >= 0; no--)
      {
            temp2 = read2 % 100;
            read2 = read2 / 100;
            message4[no] = (temp2 + 55);
      }
      for (no = 0; no < 5; no++)
      {
            message3[no] = char(message4[no]);
      }
      for (no = 0; no < 5; no++)
      {
            cout << message3[no];
      }
      cout << endl;

      write1 = 284659634659;
      write2 = 1746535356;
      write3 = 2846606029504452;
      write4 = 2242606128464459;
      write5 = 1249425548461250;
      write6 = 5749465928574644;
      write7 = 1550555060494645;
      f.write(write1);
      g.write(write2);
      h.write(write3);
      i.write(write4);
      j.write(write5);
      k.write(write6);
```

```
                        l.write(write7);
                        cout << "Sensor1 to Enduser: Server Hello
SessTick MastSecr ChangeCipherSpec Finished" << endl;

                        wait(2);

                        read1 = d.read();
                        read2 = e.read();
                        read3 = m.read();
                        read4 = n.read();
                        if (read1 == 1249425548461250 && read2 ==
5749465928574644 && read3 == 1550555060494645 && read4 ==
28464462594645)
                        {
                              for (no = 7; no >= 0; no--)
                              {
                                    temp = read1 % 100;
                                    read1 = read1 / 100;
                                    message2[no] = (temp + 55);
                              }
                              for (no = 0; no < 8; no++)
                              {
                                    message1[no] = char(message2[no]);
                              }
                              cout << "Sensor1 receives: ";
                              for (no = 0; no < 8; no++)
                              {
                                    cout << message1[no];
                              }

                              for (no = 7; no >= 0; no--)
                              {
                                    temp = read2 % 100;
                                    read2 = read2 / 100;
                                    message2[no] = (temp + 55);
                              }
                              for (no = 0; no < 8; no++)
                              {
                                    message1[no] = char(message2[no]);
                              }
                              for (no = 0; no < 8; no++)
                              {
                                    cout << message1[no];
                              }
                              cout << " ";

                              for (no = 7; no >= 0; no--)
                              {
                                    temp = read3 % 100;
                                    read3 = read3 / 100;
                                    message2[no] = (temp + 55);
                              }
```

```
                    for (no = 0; no < 8; no++)
                    {
                         message1[no] = char(message2[no]);
                    }
                    for (no = 0; no < 8; no++)
                    {
                         cout << message1[no];
                    }
                    cout << " ";

                    for (no = 6; no >= 0; no--)
                    {
                         temp = read4 % 100;
                         read4 = read4 / 100;
                         message2[no] = (temp + 55);
                    }
                    for (no = 0; no < 7; no++)
                    {
                         message1[no] = char(message2[no]);
                    }
                    for (no = 0; no < 7; no++)
                    {
                         cout << message1[no];
                    }
                    cout << endl << endl;

                    wait(25);

                    temp = 0;
                    int value[6], number;
                    std::ifstream
infile("/home/naren/Downloads/systemc-
2.3.2/Practice/Simulation/readings.txt");
                    std::string line;
                    while(std::getline(infile, line))
                    {
                         std::istringstream iss(line);
                         iss >> temp;
                         value[number] = temp;
                         number++;
                    }
                    for(number = 0; number < 5; number++)
                    {
                         o.write(value[number]);
                         wait(15);
                    }
               }

          }

     }
     sc_stop();
```

```
      }
}
```

Sensor2.h

```
SC_MODULE(sensor_2)
{
      sc_in_clk clk;
      sc_in<sc_uint<64>> a;
      sc_in<sc_uint<64>> b;
      sc_in<sc_uint<64>> c;
      sc_in<sc_uint<64>> d;
      sc_in<sc_uint<64>> e;
      sc_out<sc_uint<64>> f;
      sc_out<sc_uint<64>> g;
      sc_out<sc_uint<64>> h;
      sc_out<sc_uint<64>> i;
      sc_out<sc_uint<64>> j;
      sc_out<sc_uint<64>> k;
      sc_out<sc_uint<64>> l;
      sc_in<sc_uint<64>> m;
      sc_in<sc_uint<64>> n;
      sc_out<sc_uint<64>> o;

      SC_CTOR(sensor_2)
      {
            SC_CTHREAD(connect4, clk.pos());
      }

      void connect4();
};
```

Sensor2.cpp

```
#include <systemc.h>
#include "sensor2.h"
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>

using namespace std;

void sensor_2::connect4()
{
     wait(16);
     while(true)
     {
           sc_uint<64> read1, read2, read3, read4, write1, write2,
write3, write4, write5, write6, write7;
           int no, temp, temp2, temp3;
           read1 = a.read();
           read2 = b.read();
           read3 = c.read();
           char message1[9], message3[6], message5[6];
           int message2[9], message4[6], message6[6];

           if(read1 == 14554562604659 && read2 == 1062614956 && read3
== 5950674645)
           {
                 for (no = 6; no >= 0; no--)
                 {
                       temp = read1 % 100;
                       read1 = read1 / 100;
                       message2[no] = (temp + 55);
                 }
                 for (no = 0; no < 7; no++)
                 {
                       message1[no] = char(message2[no]);
                 }
                 cout << "Sensor2 receives: ";
                 for (no = 0; no < 7; no++)
                 {
                       cout << message1[no];
                 }
                 cout << " ";

                 for (no = 4; no >= 0; no--)
                 {
                       temp2 = read2 % 100;
                       read2 = read2 / 100;
                       message4[no] = (temp2 + 55);
                 }
                 for (no = 0; no < 5; no++)
                 {
                       message3[no] = char(message4[no]);
                 }
                 for (no = 0; no < 5; no++)
                 {
                       cout << message3[no];
                 }

                 for (no = 4; no >= 0; no--)
                 {
                       temp3 = read3 % 100;
                       read3 = read3 / 100;
                       message6[no] = (temp3 + 55);
```

```
}
for (no = 0; no < 5; no++)
{
     message5[no] = char(message6[no]);
}
for (no = 0; no < 5; no++)
{
     cout << message5[no];
}
cout << endl;
wait(2);

read1 = d.read();
read2 = e.read();
if(read1 == 125350465561 && read2 == 1746535356)
{
     for (no = 5; no >= 0; no--)
     {
          temp = read1 % 100;
          read1 = read1 / 100;
          message2[no] = (temp + 55);
     }
     for (no = 0; no < 6; no++)
     {
          message1[no] = char(message2[no]);
     }
     cout << "Sensor2 receives: ";
     for (no = 0; no < 6; no++)
     {
          cout << message1[no];
     }
     cout << " ";

     for (no = 4; no >= 0; no--)
     {
          temp2 = read2 % 100;
          read2 = read2 / 100;
          message4[no] = (temp2 + 55);
     }
     for (no = 0; no < 5; no++)
     {
          message3[no] = char(message4[no]);
     }
     for (no = 0; no < 5; no++)
     {
          cout << message3[no];
     }
     cout << endl;

     write1 = 284659634659;
     write2 = 1746535356;
     write3 = 2846606029504452;
```

```
                    write4 = 2242606128464459;
                    write5 = 1249425548461250;
                    write6 = 5749465928574644;
                    write7 = 1550555060494645;
                    f.write(write1);
                    g.write(write2);
                    h.write(write3);
                    i.write(write4);
                    j.write(write5);
                    k.write(write6);
                    l.write(write7);
                    cout << "Sensor2 to Enduser: Server Hello
SessTick MastSecr ChangeCipherSpec Finished" << endl;

                    wait(2);

                    read1 = d.read();
                    read2 = e.read();
                    read3 = m.read();
                    read4 = n.read();
                    if (read1 == 1249425548461250 && read2 ==
5749465928574644 && read3 == 1550555060494645 && read4 ==
28464462594645)
                    {
                         for (no = 7; no >= 0; no--)
                         {
                              temp = read1 % 100;
                              read1 = read1 / 100;
                              message2[no] = (temp + 55);
                         }
                         for (no = 0; no < 8; no++)
                         {
                              message1[no] = char(message2[no]);
                         }
                         cout << "Sensor2 receives: ";
                         for (no = 0; no < 8; no++)
                         {
                              cout << message1[no];
                         }

                         for (no = 7; no >= 0; no--)
                         {
                              temp = read2 % 100;
                              read2 = read2 / 100;
                              message2[no] = (temp + 55);
                         }
                         for (no = 0; no < 8; no++)
                         {
                              message1[no] = char(message2[no]);
                         }
                         for (no = 0; no < 8; no++)
                         {
                              cout << message1[no];
                         }
                         cout << " ";

                         for (no = 7; no >= 0; no--)
                         {
                              temp = read3 % 100;
                              read3 = read3 / 100;
                              message2[no] = (temp + 55);
                         }
                         for (no = 0; no < 8; no++)
                         {
                              message1[no] = char(message2[no]);
                         }
                         for (no = 0; no < 8; no++)
                         {
                              cout << message1[no];
                         }
                         cout << " ";

                         for (no = 6; no >= 0; no--)
                         {
                              temp = read4 % 100;
                              read4 = read4 / 100;
                              message2[no] = (temp + 55);
                         }
                         for (no = 0; no < 7; no++)
                         {
                              message1[no] = char(message2[no]);
                         }
                         for (no = 0; no < 7; no++)
                         {
                              cout << message1[no];
                         }
                         cout << endl << endl;
                         wait(22);

                         temp = 0;
                         int value[6], number;
                         std::ifstream
infile("/home/naren/Downloads/systemc-
2.3.2/Practice/Simulation/readings.txt");
                         std::string line;
                         while(std::getline(infile, line))
                         {
                              std::istringstream iss(line);
                              iss >> temp;
                              value[number] = temp;
                              number++;
                         }
                         for(number = 0; number < 5; number++)
                         {
                                o.write(value[number]);
                                wait(15);
                           }
                      }

                 }

            }
            sc_stop();

      }
}
```

Sensor3.h

```
SC_MODULE(sensor_3)
{
      sc_in_clk clk;
      sc_in<sc_uint<64>> a;
      sc_in<sc_uint<64>> b;
      sc_in<sc_uint<64>> c;
      sc_in<sc_uint<64>> d;
      sc_in<sc_uint<64>> e;
      sc_out<sc_uint<64>> f;
      sc_out<sc_uint<64>> g;
      sc_out<sc_uint<64>> h;
      sc_out<sc_uint<64>> i;
      sc_out<sc_uint<64>> j;
      sc_out<sc_uint<64>> k;
      sc_out<sc_uint<64>> l;
      sc_in<sc_uint<64>> m;
      sc_in<sc_uint<64>> n;
      sc_out<sc_uint<64>> o;

      SC_CTOR(sensor_3)
      {
            SC_CTHREAD(connect5, clk.pos());
      }

      void connect5();
};
```

Sensor3.cpp

```
#include <systemc.h>
#include "sensor3.h"
```

```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>

using namespace std;

void sensor_3::connect5()
{
	wait(22);
	while(true)
	{
		sc_uint<64> read1, read2, read3, read4, write1, write2,
write3, write4, write5, write6, write7;
		int no, temp, temp2, temp3;
		read1 = a.read();
		read2 = b.read();
		read3 = c.read();
		char message1[9], message3[6], message5[6];
		int message2[9], message4[6], message6[6];

		if(read1 == 14554562604659 && read2 == 1062614956 && read3
== 5950674645)
		{
			for (no = 6; no >= 0; no--)
			{
				temp = read1 % 100;
				read1 = read1 / 100;
				message2[no] = (temp + 55);
			}
			for (no = 0; no < 7; no++)
			{
				message1[no] = char(message2[no]);
			}
			cout << "Sensor3 receives: ";
			for (no = 0; no < 7; no++)
			{
				cout << message1[no];
			}
			cout << " ";

			for (no = 4; no >= 0; no--)
			{
				temp2 = read2 % 100;
				read2 = read2 / 100;
				message4[no] = (temp2 + 55);
			}
			for (no = 0; no < 5; no++)
			{
				message3[no] = char(message4[no]);
			}
			for (no = 0; no < 5; no++)
```

```
{
      cout << message3[no];
}

for (no = 4; no >= 0; no--)
{
      temp3 = read3 % 100;
      read3 = read3 / 100;
      message6[no] = (temp3 + 55);
}
for (no = 0; no < 5; no++)
{
      message5[no] = char(message6[no]);
}
for (no = 0; no < 5; no++)
{
      cout << message5[no];
}
cout << endl;
wait(2);

read1 = d.read();
read2 = e.read();
if(read1 == 125350465561 && read2 == 1746535356)
{
      for (no = 5; no >= 0; no--)
      {
            temp = read1 % 100;
            read1 = read1 / 100;
            message2[no] = (temp + 55);
      }
      for (no = 0; no < 6; no++)
      {
            message1[no] = char(message2[no]);
      }
      cout << "Sensor3 receives: ";
      for (no = 0; no < 6; no++)
      {
            cout << message1[no];
      }
      cout << " ";

      for (no = 4; no >= 0; no--)
      {
            temp2 = read2 % 100;
            read2 = read2 / 100;
            message4[no] = (temp2 + 55);
      }
      for (no = 0; no < 5; no++)
      {
            message3[no] = char(message4[no]);
      }
```

```
                    for (no = 0; no < 5; no++)
                    {
                         cout << message3[no];
                    }
                    cout << endl;

                    write1 = 284659634659;
                    write2 = 1746535356;
                    write3 = 2846606029504452;
                    write4 = 2242606128464459;
                    write5 = 1249425548461250;
                    write6 = 5749465928574644;
                    write7 = 1550555060494645;
                    f.write(write1);
                    g.write(write2);
                    h.write(write3);
                    i.write(write4);
                    j.write(write5);
                    k.write(write6);
                    l.write(write7);
                    cout << "Sensor3 to Enduser: Server Hello
SessTick MastSecr ChangeCipherSpec Finished" << endl;

                    wait(2);

                    read1 = d.read();
                    read2 = e.read();
                    read3 = m.read();
                    read4 = n.read();
                    if (read1 == 1249425548461250 && read2 ==
5749465928574644 && read3 == 1550555060494645 && read4 ==
28464462594645)
                    {
                         for (no = 7; no >= 0; no--)
                         {
                              temp = read1 % 100;
                              read1 = read1 / 100;
                              message2[no] = (temp + 55);
                         }
                         for (no = 0; no < 8; no++)
                         {
                              message1[no] = char(message2[no]);
                         }
                         cout << "Sensor3 receives: ";
                         for (no = 0; no < 8; no++)
                         {
                              cout << message1[no];
                         }

                         for (no = 7; no >= 0; no--)
                         {
                              temp = read2 % 100;
```

```
                                          read2 = read2 / 100;
                                          message2[no] = (temp + 55);
                                    }
                                    for (no = 0; no < 8; no++)
                                    {
                                          message1[no] = char(message2[no]);
                                    }
                                    for (no = 0; no < 8; no++)
                                    {
                                          cout << message1[no];
                                    }
                                    cout << " ";

                                    for (no = 7; no >= 0; no--)
                                    {
                                          temp = read3 % 100;
                                          read3 = read3 / 100;
                                          message2[no] = (temp + 55);
                                    }
                                    for (no = 0; no < 8; no++)
                                    {
                                          message1[no] = char(message2[no]);
                                    }
                                    for (no = 0; no < 8; no++)
                                    {
                                          cout << message1[no];
                                    }
                                    cout << " ";

                                    for (no = 6; no >= 0; no--)
                                    {
                                          temp = read4 % 100;
                                          read4 = read4 / 100;
                                          message2[no] = (temp + 55);
                                    }
                                    for (no = 0; no < 7; no++)
                                    {
                                          message1[no] = char(message2[no]);
                                    }
                                    for (no = 0; no < 7; no++)
                                    {
                                          cout << message1[no];
                                    }
                                    cout << endl << endl;
                                    wait(19);

                                    temp = 0;
                                    int value[6], number;
                                    std::ifstream
infile("/home/naren/Downloads/systemc-
2.3.2/Practice/Simulation/readings.txt");
                                    std::string line;
```

```
                              while(std::getline(infile, line))
                              {
                                   std::istringstream iss(line);
                                   iss >> temp;
                                   value[number] = temp;
                                   number++;
                              }
                              for(number = 0; number < 5; number++)
                              {
                                   o.write(value[number]);
                                   wait(15);
                              }
                         }

                    }

               }
               sc_stop();

          }
}
```

Sensor4.h

```
SC_MODULE(sensor_4)
{
     sc_in_clk clk;
     sc_in<sc_uint<64>> a;
     sc_in<sc_uint<64>> b;
     sc_in<sc_uint<64>> c;
     sc_in<sc_uint<64>> d;
     sc_in<sc_uint<64>> e;
     sc_out<sc_uint<64>> f;
     sc_out<sc_uint<64>> g;
     sc_out<sc_uint<64>> h;
     sc_out<sc_uint<64>> i;
     sc_out<sc_uint<64>> j;
     sc_out<sc_uint<64>> k;
     sc_out<sc_uint<64>> l;
     sc_in<sc_uint<64>> m;
     sc_in<sc_uint<64>> n;
     sc_out<sc_uint<64>> o;

     SC_CTOR(sensor_4)
     {
          SC_CTHREAD(connect6, clk.pos());
     }

     void connect6();
};
```

Sensor4.cpp

```
#include <systemc.h>
#include "sensor4.h"
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>

using namespace std;

void sensor_4::connect6()
{
      wait(28);
      while(true)
      {
            sc_uint<64> read1, read2, read3, read4, write1, write2,
write3, write4, write5, write6, write7;
            int no, temp, temp2, temp3;
            read1 = a.read();
            read2 = b.read();
            read3 = c.read();
            char message1[9], message3[6], message5[6];
            int message2[9], message4[6], message6[6];

            if(read1 == 14554562604659 && read2 == 1062614956 && read3
== 5950674645)
            {
                  for (no = 6; no >= 0; no--)
                  {
                        temp = read1 % 100;
                        read1 = read1 / 100;
                        message2[no] = (temp + 55);
                  }
                  for (no = 0; no < 7; no++)
                  {
                        message1[no] = char(message2[no]);
                  }
                  cout << "Sensor4 receives: ";
                  for (no = 0; no < 7; no++)
                  {
                        cout << message1[no];
                  }
                  cout << " ";

                  for (no = 4; no >= 0; no--)
                  {
                        temp2 = read2 % 100;
                        read2 = read2 / 100;
                        message4[no] = (temp2 + 55);
                  }
```

```
for (no = 0; no < 5; no++)
{
	message3[no] = char(message4[no]);
}
for (no = 0; no < 5; no++)
{
	cout << message3[no];
}

for (no = 4; no >= 0; no--)
{
	temp3 = read3 % 100;
	read3 = read3 / 100;
	message6[no] = (temp3 + 55);
}
for (no = 0; no < 5; no++)
{
	message5[no] = char(message6[no]);
}
for (no = 0; no < 5; no++)
{
	cout << message5[no];
}
cout << endl;
wait(2);

read1 = d.read();
read2 = e.read();
if(read1 == 125350465561 && read2 == 1746535356)
{
	for (no = 5; no >= 0; no--)
	{
		temp = read1 % 100;
		read1 = read1 / 100;
		message2[no] = (temp + 55);
	}
	for (no = 0; no < 6; no++)
	{
		message1[no] = char(message2[no]);
	}
	cout << "Sensor4 receives: ";
	for (no = 0; no < 6; no++)
	{
		cout << message1[no];
	}
	cout << " ";

	for (no = 4; no >= 0; no--)
	{
		temp2 = read2 % 100;
		read2 = read2 / 100;
		message4[no] = (temp2 + 55);
```

```
                      }
                      for (no = 0; no < 5; no++)
                      {
                           message3[no] = char(message4[no]);
                      }
                      for (no = 0; no < 5; no++)
                      {
                           cout << message3[no];
                      }
                      cout << endl;

                      write1 = 284659634659;
                      write2 = 1746535356;
                      write3 = 2846606029504452;
                      write4 = 2242606128464459;
                      write5 = 1249425548461250;
                      write6 = 5749465928574644;
                      write7 = 1550555060494645;
                      f.write(write1);
                      g.write(write2);
                      h.write(write3);
                      i.write(write4);
                      j.write(write5);
                      k.write(write6);
                      l.write(write7);
                      cout << "Sensor4 to Enduser: Server Hello
SessTick MastSecr ChangeCipherSpec Finished" << endl;

                      wait(2);

                      read1 = d.read();
                      read2 = e.read();
                      read3 = m.read();
                      read4 = n.read();
                      if (read1 == 1249425548461250 && read2 ==
5749465928574644 && read3 == 1550555060494645 && read4 ==
28464462594645)
                      {
                           for (no = 7; no >= 0; no--)
                           {
                                temp = read1 % 100;
                                read1 = read1 / 100;
                                message2[no] = (temp + 55);
                           }
                           for (no = 0; no < 8; no++)
                           {
                                message1[no] = char(message2[no]);
                           }
                           cout << "Sensor4 receives: ";
                           for (no = 0; no < 8; no++)
                           {
                                cout << message1[no];
}

for (no = 7; no >= 0; no--)
{
      temp = read2 % 100;
      read2 = read2 / 100;
      message2[no] = (temp + 55);
}
for (no = 0; no < 8; no++)
{
      message1[no] = char(message2[no]);
}
for (no = 0; no < 8; no++)
{
      cout << message1[no];
}
cout << " ";

for (no = 7; no >= 0; no--)
{
      temp = read3 % 100;
      read3 = read3 / 100;
      message2[no] = (temp + 55);
}
for (no = 0; no < 8; no++)
{
      message1[no] = char(message2[no]);
}
for (no = 0; no < 8; no++)
{
      cout << message1[no];
}
cout << " ";

for (no = 6; no >= 0; no--)
{
      temp = read4 % 100;
      read4 = read4 / 100;
      message2[no] = (temp + 55);
}
for (no = 0; no < 7; no++)
{
      message1[no] = char(message2[no]);
}
for (no = 0; no < 7; no++)
{
      cout << message1[no];
}
cout << endl << endl;
wait(16);

temp = 0;
```

```
                              int value[6], number;
                              std::ifstream
infile("/home/naren/Downloads/systemc-
2.3.2/Practice/Simulation/readings.txt");
                              std::string line;
                              while(std::getline(infile, line))
                              {
                                   std::istringstream iss(line);
                                   iss >> temp;
                                   value[number] = temp;
                                   number++;
                              }
                              for(number = 0; number < 5; number++)
                              {
                                   o.write(value[number]);
                                   wait(15);
                              }
                         }

                    }

               }
               sc_stop();

     }
}
```

Sensor5.h

```
SC_MODULE(sensor_5)
{
     sc_in_clk clk;
     sc_in<sc_uint<64>> a;
     sc_in<sc_uint<64>> b;
     sc_in<sc_uint<64>> c;
     sc_in<sc_uint<64>> d;
     sc_in<sc_uint<64>> e;
     sc_out<sc_uint<64>> f;
     sc_out<sc_uint<64>> g;
     sc_out<sc_uint<64>> h;
     sc_out<sc_uint<64>> i;
     sc_out<sc_uint<64>> j;
     sc_out<sc_uint<64>> k;
     sc_out<sc_uint<64>> l;
     sc_in<sc_uint<64>> m;
     sc_in<sc_uint<64>> n;
     sc_out<sc_uint<64>> o;

     SC_CTOR(sensor_5)
```

```
	{
		SC_CTHREAD(connect7, clk.pos());
	}

	void connect7();
};
```

Sensor5.cpp

```
#include <systemc.h>
#include "sensor5.h"
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>

using namespace std;

void sensor_5::connect7()
{
	wait(34);
	while(true)
	{
		sc_uint<64> read1, read2, read3, read4, write1, write2, 
write3, write4, write5, write6, write7;
		int no, temp, temp2, temp3;
		read1 = a.read();
		read2 = b.read();
		read3 = c.read();
		char message1[9], message3[6], message5[6];
		int message2[9], message4[6], message6[6];

		if(read1 == 14554562604659 && read2 == 1062614956 && read3 
== 5950674645)
		{
			for (no = 6; no >= 0; no--)
			{
				temp = read1 % 100;
				read1 = read1 / 100;
				message2[no] = (temp + 55);
			}
			for (no = 0; no < 7; no++)
			{
				message1[no] = char(message2[no]);
			}
			cout << "Sensor5 receives: ";
			for (no = 0; no < 7; no++)
			{
				cout << message1[no];
```

```
}
cout << " ";

for (no = 4; no >= 0; no--)
{
      temp2 = read2 % 100;
      read2 = read2 / 100;
      message4[no] = (temp2 + 55);
}
for (no = 0; no < 5; no++)
{
      message3[no] = char(message4[no]);
}
for (no = 0; no < 5; no++)
{
      cout << message3[no];
}

for (no = 4; no >= 0; no--)
{
      temp3 = read3 % 100;
      read3 = read3 / 100;
      message6[no] = (temp3 + 55);
}
for (no = 0; no < 5; no++)
{
      message5[no] = char(message6[no]);
}
for (no = 0; no < 5; no++)
{
      cout << message5[no];
}
cout << endl;
wait(2);

read1 = d.read();
read2 = e.read();
if(read1 == 125350465561 && read2 == 1746535356)
{
      for (no = 5; no >= 0; no--)
      {
            temp = read1 % 100;
            read1 = read1 / 100;
            message2[no] = (temp + 55);
      }
      for (no = 0; no < 6; no++)
      {
            message1[no] = char(message2[no]);
      }
      cout << "Sensor5 receives: ";
      for (no = 0; no < 6; no++)
      {
```

```
                              cout << message1[no];
                        }
                        cout << " ";

                        for (no = 4; no >= 0; no--)
                        {
                              temp2 = read2 % 100;
                              read2 = read2 / 100;
                              message4[no] = (temp2 + 55);
                        }
                        for (no = 0; no < 5; no++)
                        {
                              message3[no] = char(message4[no]);
                        }
                        for (no = 0; no < 5; no++)
                        {
                              cout << message3[no];
                        }
                        cout << endl;

                        write1 = 284659634659;
                        write2 = 1746535356;
                        write3 = 2846606029504452;
                        write4 = 2242606128464459;
                        write5 = 1249425548461250;
                        write6 = 5749465928574644;
                        write7 = 1550555060494645;
                        f.write(write1);
                        g.write(write2);
                        h.write(write3);
                        i.write(write4);
                        j.write(write5);
                        k.write(write6);
                        l.write(write7);
                        cout << "Sensor5 to Enduser: Server Hello
SessTick MastSecr ChangeCipherSpec Finished" << endl;

                        wait(2);

                        read1 = d.read();
                        read2 = e.read();
                        read3 = m.read();
                        read4 = n.read();
                        if (read1 == 1249425548461250 && read2 ==
5749465928574644 && read3 == 1550555060494645 && read4 ==
28464462594645)
                        {
                              for (no = 7; no >= 0; no--)
                              {
                                    temp = read1 % 100;
                                    read1 = read1 / 100;
                                    message2[no] = (temp + 55);
```

```
}
for (no = 0; no < 8; no++)
{
     message1[no] = char(message2[no]);
}
cout << "Sensor5 receives: ";
for (no = 0; no < 8; no++)
{
     cout << message1[no];
}

for (no = 7; no >= 0; no--)
{
     temp = read2 % 100;
     read2 = read2 / 100;
     message2[no] = (temp + 55);
}
for (no = 0; no < 8; no++)
{
     message1[no] = char(message2[no]);
}
for (no = 0; no < 8; no++)
{
     cout << message1[no];
}
cout << " ";

for (no = 7; no >= 0; no--)
{
     temp = read3 % 100;
     read3 = read3 / 100;
     message2[no] = (temp + 55);
}
for (no = 0; no < 8; no++)
{
     message1[no] = char(message2[no]);
}
for (no = 0; no < 8; no++)
{
     cout << message1[no];
}
cout << " ";

for (no = 6; no >= 0; no--)
{
     temp = read4 % 100;
     read4 = read4 / 100;
     message2[no] = (temp + 55);
}
for (no = 0; no < 7; no++)
{
     message1[no] = char(message2[no]);
```

```
                                }
                                for (no = 0; no < 7; no++)
                                {
                                      cout << message1[no];
                                }
                                cout << endl << endl;
                                wait(13);

                                temp = 0;
                                int value[6], number;
                                std::ifstream
infile("/home/naren/Downloads/systemc-
2.3.2/Practice/Simulation/readings.txt");
                                std::string line;
                                while(std::getline(infile, line))
                                {
                                      std::istringstream iss(line);
                                      iss >> temp;
                                      value[number] = temp;
                                      number++;
                                }
                                for(number = 0; number < 5; number++)
                                {
                                      o.write(value[number]);
                                      wait(15);
                                }
                          }

                    }

            }
            sc_stop();

      }
}
```