Inter-Device Authentication Protocol for the Internet of Things

by

Preethy Wilson
B. Tech, Cochin University of Science and Technology, 2012

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Inter-Device Authentication Protocol for the Internet of Things

by

Preethy Wilson
B. Tech, Cochin University of Science and Technology, 2012

Supervisory Committee

Dr. Gebali. Fayez, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Sima. Mihai, Member
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Gebali. Fayez, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Sima. Mihai, Member
(Department of Electrical and Computer Engineering)

ABSTRACT

The Internet of things (IoT) recently blossomed remarkably and has been transforming the everyday physical entities around us into an ecosystem of information that will enrich our lives in unimaginable ways. Authentication is one of the primary goals of security in the IoT and acts as the main gateway to a secure system which transmits confidential and/or private data.This thesis focuses on a Device-to-Device Mutual Authentication Protocol, designed for the smart home network, which is an essential component of communication in the Internet of Things(IoT).

The protocol has been developed based on asymmetric cryptography to authenticate the devices in the network and for the devices to agree on a shared secret session key. In order to ensure the security of a communications session between the devices, the session keys are changed frequently - ideally after every communication session. The proposed scheme has been programmed in HLPSL, simulated and its efficiency verified using the SPAN/ AVISPA tool. When SPAN substantiates the protocol simulation and the attacker simulation, the back-ends of the AVISPA tool verifies the safety and security of the proposed authentication protocol. The thesis also evaluates the protocol's security against the attacks successful against protocols proposed by other researchers.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

First and foremost, I express my sincere gratitude to God for His blessings and for enabling me to accomplish this remarkable feat in my career.

I would like to thank Dr. Fayez Gebali for his support and guidance throughout my program and during the process of writing this MASc. thesis. He has imparted the fundamental skills of our discipline and, more importantly, he has taught me how to further my skills independently.

My gratitude also goes to Dr. Sherif Saad, whose vision, expertise and knowledge, with Dr. Fayez Gebali's supervision, guided me to complete my thesis in an independent and productive manner. He helped me explore ideas, to critically evaluate my work, and to express my results clearly.

Finally, I would like to thank my family and my friends, who were a very crucial part of this achievement, for their prayers, unfailing personal support and encouragement throughout my studies and the masters thesis.

DEDICATION

To

John Wilson (father)
Lizy Wilson (mother)
Deepthi Wilson (sister)

LIST OF ABBREVIATIONS

AVISPA	Automated Validation of Internet Security Protocols and Applications
CLAtSe	Constraint-Logic-based Attack Searcher
CPU	Central Processing Unit
D2D	Device-to-Device
DoS	Denial-of-Service
DTLS	Datagram Transport Layer Security
EAP	Extensible Authentication Protocol
ECC	Elliptic Curve Cryptography
IoT	Internet of Things
GBA	Group-based Authentication
HLPSL	High-Level Protocol Specification Language
ID	Identification
IF	Intermediate Format
IP	Internet Protocol
IPv6	Internet Protocol Version-6
KDC	Key Distribution Center
LAN	Local Area Network
LLN	Low-Power Lossy Networks
M2M	Machine-to-Machine
MAC	Message Authentication Code
MFA	Multi-Factor Authentication
MTCD	Machine-type Communication Device
OFMC	On-the-fly Model-Checker
OTP	One-Time Password
PDA	Personal Digital Assistant
RFID	Radio-Frequency Identification
SATMC	SAT-based Model-Checker
SPAN	Security Protocol Animator
TA4SP	Tree Automata based on Automatic Approximations for the Analysis of Security Protocols
VPN	Virtual Private Network

Chapter 1

Introduction

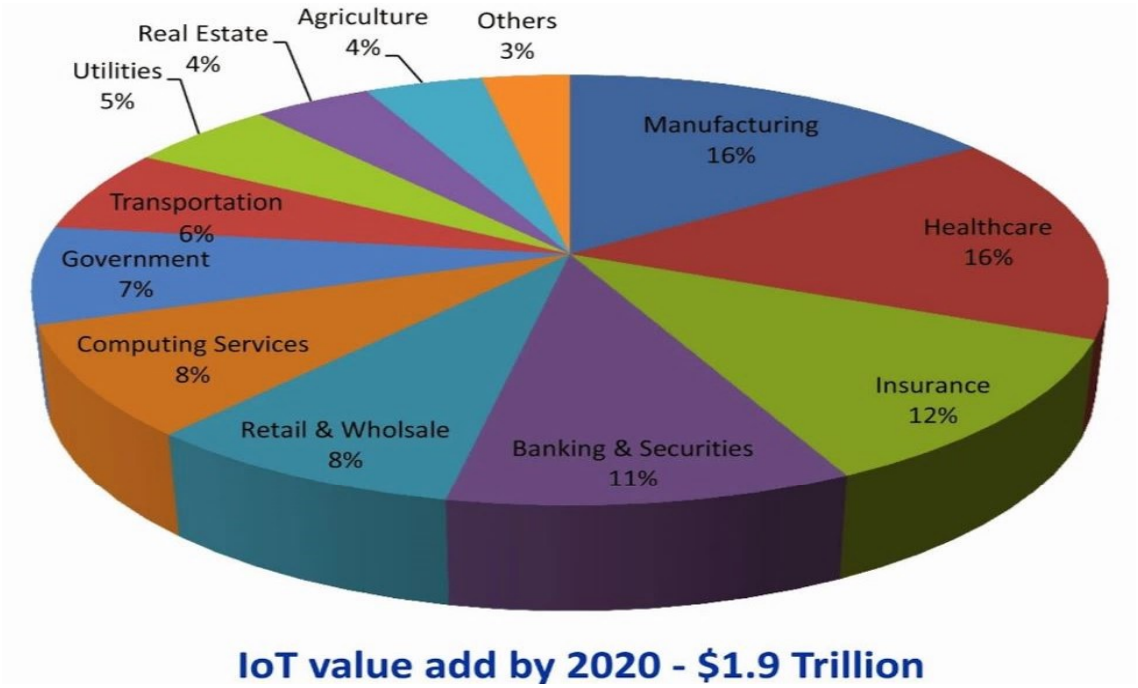
This chapter briefs the concepts of the Internet of Things and authentication. It also explains the smart home environment in the IoT and also discusses an architecture for the smart home network, taking into consideration the constraints of the devices used in the IoT.

1.1 Internet of Things and Authentication

The Internet of things (IoT) recently blossomed remarkably which prompted swarms of startups, many major companies and organizations to invest their energy, time and resources in this growing field. The name Internet of Things, coined by Kevin Ashton, originated in the Auto-ID Lab Centre at Massachusetts Institute of Technology in 1999; with the aim of making an object sense and process information without human intervention. In a 1999 article for the RFID Journal, Ashton wrote:

"If we had computers that knew everything there was to know about things - using data they gathered without any help from us – we would be able to track and count everything, and greatly reduce waste, loss and cost. We need to empower computers with their own means of gathering information, so they can see, hear and smell the world for themselves, in all its random glory [1]".

Since then, the IoT has been transforming the everyday physical entities around us into an ecosystem of information that will enrich our lives in unimaginable ways. It is an integration of a variety of existing and new technologies. It incorporates a large number of assorted end systems enabling mutual communication and open access to data between them.



Source: Gartner

Figure 1.1: Verticals in the IoT.

"From refrigerators to houses to parking spaces, the IoT is bringing more and more things into the digital fold every day, which will likely make the IoT a multi-trillion dollar industry in the near future"— as conveyed by PricewaterhouseCoopers report [2].

Popular devices such as fitbits, pacemakers, insulin pumps, Apple watches, home thermostats and smart TV can be networked together to enrich the end user's life. Machine-to-machine communication in industrial environments, automobile collision avoidance/ detection, driverless vehicles and smart grid devices that monitor electrical consumption are also in demand now. These devices enhance connectivity and thereby improve economies, quality of life, environments and even health. IoT can be applied to various domains or verticals as shown in Fig.1.1.

According to ISO/IEC 27002, authentication is the act of establishing, or confirming something (or someone) as authentic. Authentication is an important operation at the core of the IoT security framework. It is the process of validating a device's identity in communication and ensuring the reliability of the origin of communication. By enabling server and user authentication, we are improving the trust factor in IoT

communications and also ensuring information security. Authentication is one of the primary goals of security and acts as the main gateway to a secure system which transmits confidential and/or private data.

Authorization is another component in IoT security which builds on the information provided during authentication. This process assumes that authentication of devices has already been carried out in the network and is mainly concerned with assigning each user's access privileges.

From an end user's perspective, both these processes occur at the same time but these concepts are critical when designing the security framework of an application. Both authentication and authorization need to be in place to establish a secure relationship between IoT resources before any communication of data or information occurs. Multi-factor Authentication (MFA) aims to create a layered defense system that combines two or more factors from independent categories of credentials such that unauthorized users are required to make strenuous effort to gain access to a target. MFA has been widely used in recent years to ensure secure authentication in systems as it provides an extra layer of protection.

Consider a Fitbit wristband, which gives users feedback on their daily activity. A Fitbit is a tiny connected computer, tightly bound to a particular user. When in use, a Fitbit, and other similar devices, could facilitate authentication of the user as they access applications, devices or cloud services. The Nymi device takes the idea one step further by adding a biometric authentication of the user; it won't make the keys it stores available for authentication until it validates the wearer's electrocardiogram against the stored template.

Beyond wearables, there lies a category of passive authentication that can be enabled by other devices that will soon surround us. Current fraud detection systems use the IP address of the computer to identify attacks initiated from a locale other than expected. The potential for capturing such context through IoT devices, such as a home automation motion detector could report that the house was empty. Thus, preventing an attack of a WiFi network needs to be considered. The opportunity and potential for authentication in the future is remarkable [3].

1.2 Background Research

Several alternatives have been proposed in the past for authentication in IoT, keeping in mind the constraints on the devices involved. Constrained devices use networks with limited bandwidth, limited processing power, and limited battery and so on. The methods proposed take into consideration one or more of these constraints but may not be suitable on the whole.

Shivraj et al. rely on the two-factor One Time Password (OTP) technique developed using a lightweight identity-based Elliptic Curve Cryptography scheme [4] and was shown to be more efficient and secure than other existing methods since the Key Distribution Center (KDC) neither requires any key storage nor stores private and public keys of devices, it only stores their IDs. Fewer resources were needed, according to this protocol, without compromising security itself. The proposed methodology does not tackle the case where a device wants to control another device in another gateway and one using a different security scheme. In [5], Crossman and Liu have proposed a new model for user or device verification by making the process user-centric. In this new verification scheme, called Smart Two Factor Authentication, after the user enters the password correctly, instead of a verification code, the user has to provide a security token like a smart card as the second authentication factor to retrieve a dynamic encryption key. The user has complete control of all their resources which saves them from worrying if the manufacturers database is hacked into and also saves the manufacturer any national security concerns. Since password authentication often rely on users being able to recall complex information, an emerging digital memories concept was proposed in [6], which involves the creation of a repository of memories specific to individuals. This scheme integrates user identity assurance into authentication schemes and one factor in this uses challenges from users'own digital memories. Here, the possibility of attacks are reduced as all the factors cannot be hacked or known by anyone other than the user. This methodology complicates phishing attacks, social engineering and shoulder surfing for attackers and the complexity can be varied based on the type of service being used. Yao et al. revised Nybergs fast one-way accumulator to design a message authentication code (MAC) based multicast authentication mechanism for small-scale IoT applications, which would be very efficient for resource constrained devices [7]. Multicast offers efficient communication in data sensing layer and network access layer of IoT.

A smart home technology enables all home appliances to be automatically controlled and monitored remotely. The mechanism proposed by the authors in [8] optimizes the interoperability among the isolated IoT domains and addresses security and privacy concerns in the IoT paradigm. The proposed lightweight authentication and authorization mechanisms secure certain assets of smart grid environments. The approach is based on EAP over LAN (EAPOL) which burdens the constrained device as they have to implement and execute EAPOL in addition to DTLS. [9] presents a Voiceprint and Internet based mobile authentication system consisting of two different subsystems. Raspberry Pi is taken as a control center of this subsystem and is connected to the mobile phones through a VPN, which provides a secure communication network. The authors also proposed a dynamic threshold method to calculate an user-specific score threshold of voiceprint verification based on Gaussian Mixture Model-Universal Background Model. After initial authentication, users read out the text which is different from the text for enrollment printed on screen. This prevents counterfeiters from using recording of the text for enrollment to obtain a legal identity. Although, the voiceprint verification is computationally elaborate and security is rendered at the expense of performance.

Authors in [10] implemented a low cost symmetric payload embedded key based robust authentication and key management on CoAP, Constrained Application Protocol. They presented a lightweight security verification scheme with key protection management to develop a secure channel for Vehicle Tracking System. This lightweight protocol eliminates expensive handshaking for reliability and is based on information about the vehicle circumstances.

Since IoT works in heterogeneous networking environments, Chu et al. proposed an authentication scheme based on Elliptic Curve Cryptography where the elliptic curve public parameters are calculated by the base station in the initialization phase [11]. Although parameters are calculated at the base station, extensive communication of values is required after computation. It also requires an offline/ predeployment phase to setup the network. In [12], authors described how the components involved can interact with each other to carry out an authentication process in different IoT cloud scenarios. In this, only secured IoT devices can be accepted for joining an IoT Cloud provider. After the joining process, different types of authentication are considered for Manufacturer, Basic user and Advanced user access. Manufacturers can periodically access the IoT device for firmware updates, config-

uration setup, bug fixing, and other administrative tasks. Basic users access IoT devices through the Cloud IoT Platform, whereas Advanced Users have direct access to IoT devices. However, in order to perform such authentications, users have to authenticate themselves by means of the IoT Cloud Provider. The authors of An OAuth based Authentication Mechanism for IoT Networks, realize an authorization management for the RESTful WebService API access based on the OAuth2.0 authorization framework [13]. When IoT devices need to be accessed by users, to avoid communicating through cloud infrastructure for every request, security infrastructure is stored outside the IoT device. This makes it less vulnerable to physical attacks. In this case, access to device resources is accomplished directly, or in a remote access scenario through the cloud infrastructure [14].

Group-based authentication (GBA) was believed to be the most appropriate solution for device authentication. The simulation results by Su et al. in [15] confirmed that GBA cannot improve performance in several cases. The method proposed reduces traffic and congestion by reducing the number of requests from MTCDs to MTCS. If any MTCD joins/ leaves a group in GBA, it will be challenging to retain security.

1.3 Smart Home Environment

The consumer/retail vertical in the IoT is driven by technological and product innovations across various applications and devices. Many connected home concepts have proved to be successful applications in the IoT industry. Everything connected in the local network can freely communicate with each other. These devices may use a back-end cloud service to monitor usage or allow users to remotely control these systems. Users can access this data or control their device through a mobile application or web portal. The convenience offered led to an increased demand which has further propelled the growth of consumer/retail vertical in the IoT industry. Of the many different smart home devices available on the market today, some devices gaining popularity are [16]:

1. Smart smoke detectors
2. Surveillance IP cameras
3. Smart thermostats

4. Smart locks
5. Smart light bulbs
6. Smart energy management devices
7. Smart hubs
8. Security alarms
9. Smart Entertainment systems (smart TV, TV set-top boxes, etc.)
10. Network attached storage (NAS) devices

Among the many applications of the IoT, the consumer/retail sector shows relatively more advancement in the coming years. Technical leaders have predicted that the greatest potential revenue growth for the IoT in the next three years is in consumer and retail markets (22%). The IoT/M2M is also expected to see significant revenue growth in technology industries (13%), aerospace and defense (10%), and education (10%).

A smart home pertains to a home which has an automated system for monitoring temperature, doors, energy saving, household appliances, devices communicating among themselves without human interference, monitors and responds to patterns in family home entertainment and so on. Since all home devices are interconnected, like light bulbs, smart locks, refrigerator, television, laptop, thermostats,etc, safety and privacy is of paramount importance. If a single device is compromised, an attacker can use that device as a pivot point and get into the network the device is a part of and eventually get access to the resources/ assets contained therein. These assets/ resources could be personal information, identity information, banking information, credit card details or security credentials or getting access to controlling another system, which can be disastrous.

In 2014, Hewlett-Packard Co. Fortify on Demand released its Internet of Things State of the Union Study, revealing 70 percent of the most commonly used Internet of Things (IoT) devices contain serious vulnerabilities and were vulnerable to hacks, either through weak passwords or unencrypted connections. Mr. Ted Harrington's, executive partner with Baltimore-based Independent Security Evaluators (ISE), team decided to test the 13 most popular home routers on the market. In a connected

home, the router is the central hub that provides the wireless data and ISE expected to break into around 30% of the top-rated and top-selling routers on Amazon or Best Buy. Inadvertently, they broke into 100% of them [17]. Since there is no anti-virus for routers, a hacked router can give the hacker access to the home network and cause a computer to download malicious software.

Here, we discuss an architecture, for smart homes, using hybrid solutions to meet the IoT requirements.

1. Computing - Enabling the IoT

   Although the fundamental concept of cloud computing originated in the 1950s, when virtualization was introduced by corporations with excess computing capacity; it wasn't until the past decade or so that cloud computing started to develop into the behemoth we know today. The computing industry gravitated towards centralized cloud services with ginormous data centers primarily because they are easier to manage. The detrimental effect of malware/ data breaches on computers in recent times turned cloud computing into an even more attractive paradigm. Cloud computing is an acceptable and beneficial solution only if we can be assured of the availability of full, uninterrupted access to the cloud server which is capable of processing and transmitting data quickly to the end device. When the IoT devices grew to exchange and accumulate big data constantly, we came to the realization that lack of bandwidth was a significant hurdle to completely depending on the cloud. Compounding this challenge, the IoT devices often require very rapid and real-time analysis of data.

   In order to bridge the gap in technological constraints (bandwidth, time constraints and scalability issues), edge computing was introduced as the breakthrough solution. Data is collected and processed at local computing devices rather than in the cloud or a remote data center. In this localized processing, the data will not be frequently sent to the centralized server for processing, but rather, each device on the network will do its own processing and then send the data. Although this ensures real-time analysis of data and relatively better security, it demands all the edge devices to be computationally powerful and have a higher memory capacity.

Fog Computing, a term coined by Cisco, aims to boost the IoT capability by extending the cloud to the resources that produce relevant data. This paradigm has devices known as fog nodes that can be deployed anywhere Internet and network connection are present. Fog nodes can be any device, like embedded servers, switches, industrial controllers or routers, that has computing capability, network connectivity and storage capacity. Fog platforms are dense computing architectures that push intelligence down to the local area network level where edge analystics pushes intelligence, computing capability and processing power directly into the devices at the edge of the network.

2. Communication

   The IoT ecosystem is required to coordinate various types of communication in its network which include:

   (a) Device-to-device (with/ without human intervention), device-to-user and vice versa, device-to-storage (distributed storage like the cloud) [18].

   (b) Intradomain (within the same network) or interdomain (across heterogeneous networks) [18].

   (c) Communication modality: Single-hop or multi- hop coomunication. In single-hop, the devices communicate with each other via a network infrastructure (an access point or a base station) whereas in multi-hop communication, devices relay information for each other to achieve end-to-end communication between a source and destination device [19].

3. Gateway

   The gateway is considered to be the core network element within the IoT. It facilitates the communication and data exchanges between devices. It acts as an interface and can handle incoming and outgoing traffic of the constrained and unconstrained networks. Gateways facilitate communication between two computers using different protocols. Protocol translation can be a significant problem when trying to connect products by different manufacturers or different version of products from the same manufacturer. The lack of uniform stadardization in communication protocols necessitate gateways understanding the addressing format between these networks.

From this discussion, it can be seen that although Fog Computing merely functions as a more selective way of ascertaining which data becomes centralized and which stays local, the limitations and advantages of each paradigm in computing need to be weighed depending on the application implemented. The architectures need to consider even the difference in communication protocols used in device-to-device communication, device-to-user communication and device-to-storage communication.The fog nodes and gateways need to have a protocol that lets them operate efficiently when the devices and end-points are added or removed from the network. Gateways have limited capacity and expandability. Configuration and programming of devices through the gateway is generally not possible. In some cases, routers are used which merely forward a message and changes only the address in case of similar networks. In disimilar networks, although the content and binary representation of the message doesn't change but it performs addressing, packet formatting and signalling changes as the message passes through the router. Some real life problems like these have not been addressed in literature. Here, we propose an IoT architecture particularly for the Smart Home network as shown in Fig.1.2. This architecture shows few home networks but this can be extended to include many home networks. Each home network contains multiple IoT devices in it.

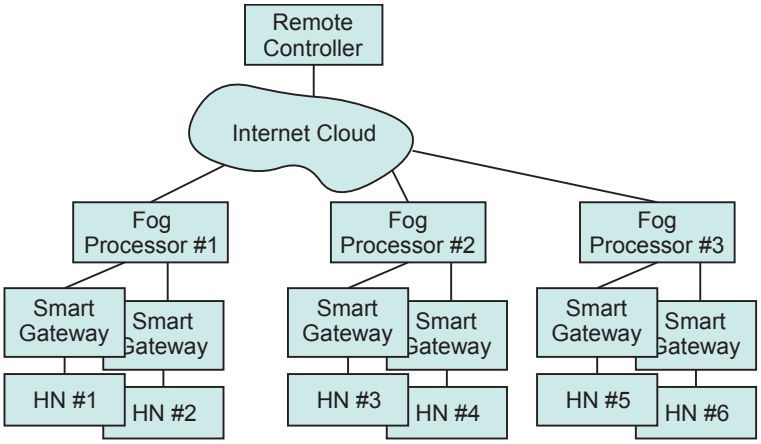


Figure 1.2: Architecture of the Smart Home Network in the IoT

where, HN refers to the Home Network.

1.4 Contributions

This thesis introduces a Device-to-Device Authentication protocol that functions without human interaction, and is proposed to be used specifically for the Smart Home IoT. Following are its contributions:

1. Propose an authentication scheme to be used for secure data communication in the IoT. The protocol is streamlined for the Smart Home networks. We use asymmetric encryption, through the use of public and private keys, to exchange data between the devices securely and eventually agree on a secret session key to be shared between the two devices. This key will be changed after every session ensuring that the data exchange will be secure every time.

2. Comparison of the authentication techniques proposed by researchers based on the authentication methodology, the evaluation model used and the security attacks they withstand. This comparative analysis showed that many proposals do not take into consideration mutual authentication between the devices involved nor do they use multiple authentication levels. Majority of the protocols considered in this section even require additional hardware.

3. The protocol is modelled in HLPSL and the algorithm verification is performed using SPAN/AVISPA tool to prove the security of the protocol against the attacks that the tool checks for.

4. Many papers discussed in the comparative analysis only perform theoretical evaluation. Here, we perform simulation and formal evaluation of the proposed protocol against attacks. In addition, we also perform theoretical evaluation against multi-protocol attack and reflection attack.

1.5 Organisation of the Thesis

The organisation of the thesis is as follows:

In Chapter 2, we present a comparative analysis of the protocols and schemes proposed by other researchers in this field. The comparison is based on the authentication methodology, the evaluation model used and the security attacks they withstand. From this analysis, it can be seen that mutual authentication is not commonly considered by the protocols discussed and some require additional hardware.

We also present the difference between traditional and the IoT authentication and the metrics that are used to evaluate an IoT authentication scheme.

In Chapter 3 we explore the phases in the lifecycle of an IoT device from the manufacturing phase to the functioning phase and explain them based on our protocol. This chapter includes a brief discussion which gives an outline to the design procedure explained in the next chapter.

In Chapter 4 we introduce the AVISPA security tool, explain its functionality, working and the back-end tools it offers. We explain the HLPSL syntax and show how programming in HLPSL can be done using some examples. We also illustrate the design for modeling the protocol in AVISPA. Programming the protocol first in A-B notation makes it easier to translate into the HLPSL notation. We also include the HLPSL implementation of the protocol, the simulation and the verification results. The formal verification process was performed using the back-end tools OFMC and Cl-ATSE which summarized that our protocol is safe.

In Chapter 5 we evaluate and discuss the security of our protocol against the attacks proposed by other researchers such as Multi-protocol attack and also Reflection attack. This theoretical evaluation process shows our protocol to be safe against these attacks also.

In Chapter 6 we conclude the thesis and show the future research possible in this area.

Chapter 2

Comparison of Authentication Techniques

In this chapter, we explain the need for a secure authentication protocol in the IoT. We also classify and compare the authentication techniques proposed by other researchers in this field and also list the metrics required to evaluate the quality and effectiveness of an authentication scheme.

2.1 Motivation/ Need for secure authentication

Internet usage, online applications and eventually the IoT are experiencing a spectacular growth. Many devices, from thermostats to security cameras are now part of the IoT, which means they can be connected to the web and can be accessed through a smartphone. Hackers have the potential to get into a wireless thermostat or invade the smart TV and from there, gain access to some other network putting the users to massive security risks.

The number of internet users is also increasing exponentially. This rapid increase and gravitation towards internet usage is attributed to the simplicity, accessibility and easy availability. As the consumer space increases, so does online crime rates and threats. Since IoT is connected to the internet for a majority of the time, it gives fraudsters ample opportunity to execute an attack with the sophisticated methodology available today. The possibility of frauds/attacks is higher and there is a high probability of these attacks going unnoticed.

Gone are the days when security breaches were restricted to email — password hacking. Current threats include hackers or intruders crashing airplanes by hacking into the networks, remotely disabling cars, remote murder through hacked medical devices, accessing home automation through hacked thermostats and hacked financial accounts [20]. Online fraud and hacking has become a major source of revenue for criminals all around the world. The motivation can be for monetary assets, business gains or out of mere curiosity. This has caused the detection and prevention of such activities a top priority.

All the functions that can be performed offline have an online counterpart today. This extends from email access, social networking, etc. to managing funds and transactions online. Strong authentication is required in cases where an user's internal network needs to be protected from unauthorized access, when users need to connect from remote locations, when an employee is accessing sensitive business data or when an user wants to implement a secure physical access solution. There will always be a trade off between security and convenience. Nevertheless, a reasonable balace between security and ease of access needs to be implemented in all functional areas of the IoT. For example; users are periodically prompted to change their authentication details which may coerce them into pulling out their authentication card. Once this is done they can rest assured that their information is better protected by those occassional inconveniences. All these applications highlight the importance of authentication as the foremost gateway to a secure connection and communication of data.

2.2 Difference between traditional and the IoT authentication

The IoT is often deployed differently than the standard internet. Consumers around the world are getting comfortable sharing, storing and accessing their private/ confidential information online. To ensure security for these applications, traditional methods such as usernames, passwords and fact-based questions are inconvenient and insufficient. Password-based authentication connection is the most popular yet one of the most insecure methods and is not suitable for computer networks. Passwords communicated over a network can be intercepted by hackers or eavesdroppers and can be ultimately used to impersonate the user.

Traditional authentication technologies in wireless networks include public key or private key based authentication, certificate based authentication, random key distribution or hash function based authentication methodologies. However, most traditional authentication is based on human credentials like biometrics, username-password combinations, some personal information or job role at an organization. In the IoT, human intervention is not included in the authentication process and the identity information depends on the machine/ device that is involved. This may mean some hardware information or IP details or MAC address.

Traditional authentication component was based on traditional role-based access control (RBAC), but in the IoT world, the attributes of a node or IoT object made more sense which makes attribute-based access control (ABAC) more suitable. RBAC controls access based on the users' roles within the system and on rules stating what access is allowed for what users in given roles. ABAC provides access to users based on who they are rather than what they do. However, IoT identity has many different contexts so a centralized solution will not be feasible. A multi-context aware mechanism is the need of the hour [21]. At every point during this authentication it should be clear who is requesting access, who is granting access, what access is requested, if it has the necessary privileges, the location, etc. Traditional authentication never had to get all these security details when access requests are issued.

In IoT, possible communications are device to device, device to human, human to device which support heterogeneous entities and networks. As devices have no prior knowledge about the other entities they are interacting with and no SSL communication, the possibility of eavesdropping is very strong. Ubiquitous data collection makes the IoT a data driven economy, but having little to no regulation on how this data is distributed, stored or accessed can be very harmful to the users. Physical limitations of the IoT has contributed to implementing security protection software or anti-virus on these devices nearly impossible.

Conventional authentication always results in overhead to the server and is a very time-consuming procedure at the end user machine. The devices connected in the IoT generally have limited memory, so the authentication methodology that will be implemented has to consider low memory capacity as the highest priority. The processing power or CPU power in these machines are much lower than what traditional networks deal with. Too much power for validation or authentication

operations doesnt work well with IoT. Many IoT networks are deployed on low-power lossy networks (LLN) while others have highly dynamic topologies depending on the application, like medical devices, home automation or vehicular networks. According to Cisco, LLNs are a class of network in which both the routers and their interconnects are constrained: constraints on processing power, memory and energy (battery), and their interconnects are characterized by high loss rates, low data rates and instability. In the case of mobile sensors or wearable devices, resource conservation is a serious aspect. LLN requires nodes be autonomous and conserve energy to enable nodes to enter power-saving modes.

The devices are also left unattended for extended periods of time. This increases the possibility of a hacker or intruder targeting the security mechanisms of these connected machines. These attacks may go unnoticed since users rarely look into the network connection and devices or change passwords/ authentication details. This will lead to conflict of interest since they are detrimental to user privacy and even national security. Increased mobility is another important factor. Users would like to access the devices from everywhere on their cellphone or PDA or desktop at home. This need also has to be addressed in the IoT security requirements.

In addition to offering convenience, the IoT authentication systems need to ensure enhanced security features, flexibility, scalability and privacy to their consumers.

2.3 Quality Metrics to Evaluate the IoT Authentication Scheme

According to National Institute of Standards and Technology (NIST), the concept of security metrics is summarized as a tool facilitating decision-making and improving the systems'performance for a specific organization or situation. The Federal Government of the United States requires that the security metrics must be integrated in the security programs of a government agency [22]. Security metrics are simply a standard or system of measurement for measuring security, specifically measuring an organizations security posture. If an issue/ problem cannot be measured, it cannot be improved either. Metrics contibute towards identifying the strengths and the weaknesses in the processes of an organization's cybersecurity program. Threat detection and risk metrics have often been considered by security professionals to be the top

indicators of a security program.

Effective metrics are often referred to as SMART, i.e. specific, measurable, attainable, repeatable, and time-dependent. To be truly useful, metrics should also indicate the degree to which security goals are being met and drive actions taken to improve an organization's overall security program [23]. The following metrics should be taken into account when developing the IoT:

1. Scalability: The scheme implemented must be able to accept new devices and the IoT architecture should be able such that more devices can be added to the current design. When new technologies are introduced, the current design should be able to adapt and easily incorporate the new modifications. The IoT architecture implemented should be such that it can be expanded in an incremental fashion to support increasing numbers of devices or growing data volumes [24].

2. Confidentiality and Data Integrity: Confidentiality ensures that an un-authenticated party does not get access to or examine the data. Data integrity guarantees that the data which is received by the receiver has not been modified after being send by the sender [25]. The architecture must support strong encryption schemes to safeguard data confidentiality and protect intellectual property.

3. Privacy: Securing the communication from the end device to the cloud or network is of utmost importance to the privacy of the devices ownership and data The information gathered should be stored in secure storage area and the transmission or communication system shall be assured of the data not being tampered with.

4. Security protocols effectiveness: Most security protocols provide a reasonable security but have several limitations. The scheme inplemented must give assurance to the users that the security protocols are effective and will be updated on a regular basis.

5. Reliability: Intelligent systems are employed for mission-critical applications, emergency services and medical applications where system downtime can result in diminished productivity, dissatisfied customers, lost revenue, loss of life or property and even cause significant environmental or health hazards [24].

6. Maintainability: This refers to addressing how the system will be maintained and serviced once it has been dispatched to consumers over the course of its life. The architecture proposed must allow making specified modifications to a component implementation in future and also enable correction of bugs and other hacks without much effort from the consumer's side.

7. Threat/Intrusion Detection, Deflection and Prevention: Amateur hackers, rival corporations, terrorists and even foreign governments have the motive and capability to carry out sophisticated attacks against computer systems. Threat/ Intrusion detection system automates the process of monitoring computers or networks for unauthorized activity. Once an unauthorized entry or modification or any suspicious activity is detected by the system, it responds by either displaying an alert or sending the required information to an administrator. Deflection is when the system responds by immediately counteracting against the attack and removing malicious content from the system, thus ensuring that it does not affect the functionality of the device nor does it put confidential and private data of the users at risk. Prevention of similar attacks can be done by reconfiguring security controls in the system like firewalls or the router to block future attacks, reconfigure security and privacy controls to filter such attacks, and block threatening packets in future [26].

8. Response Time: Some IoT systems are employed for time-critical applications which requires advanced, real-time analytics. Geo-fencing applications are those that tie digital services to real-time locations. Smart cars and self-driving vehicles depend a lot on tracking its location in time and space in real-time. When such improvements open up new areas for revenue in the automotive industry, even a slight delay will result in significant consequences and may also lead to loss of life.

9. Adherence to government policies: They must support cohesive identification, authentication and authorization capabilities to establish trust, govern access to resources, and ensure compliance with governmental regulations and corporate policies.

10. Organizational relevance: The extent to which the metric is linked to the factors that matter the most to senior management like organizational risk management, asset, threat, or vulnerability relevant to the organization [27].

11. Return of Investment (ROI): Periodic updates need to be incorporated into the design to keep up with the everchanging environment in the IoT and to ensure improved security. This metric should be able to show how important it is for the organization, show cost savings that will follow and the loss prevention that is attained in return for the expenses incurred by the organization in this regard.

2.4 Classification of Authentication Techniques

Authentication techniques can be classified in two ways. In the first way, the classification is done according to how the authentication process is performed - centralized versus distributed and hierarchial versus flat. The second way is to classify the techniques according to several attributes that reflect the general characteristics of the authentication process.

Table 2.1 classifies the the authentication techniques into:

1. Centalized: users/ devices connect to a centralized server to provide the required credentials like an authentication server or an ID-provider or any trusted third party.

2. Distributed: components communicate and coordinate between the communicating peers independently to achieve a common goal.

3. Hierarchial: uses a hierarchial authentication system in which users shall be granted access to the available information depending on the user's access rights.

4. Flat: authentication process is performed without hierarchy being taken into consideration.

Table 2.1: Classification of Authentication Techniques based on How the Authentication Process is Performed.

	Hierarchial	Flat
Centralized	[28], [12], [15]	[4], [9], [13], proposed
Distributed	[7]	[11], [5], [6], [10], [14]

Table 2.2 classifies the authentication techniques according to the characteristics of the authentication process into [29]:

1. Two-way authentication: This depends on whether mutual authentication or one-way authentication is perfomed.

2. Additional hardware required: If additional hardware is required for the authentication process to be accomplished.

3. Multiple credentials: Multiple credentials are required to be verified in several levels in order to confirm the user's or device's identity.

4. Multiple authentication levels: Multiple levels of authentication are required, each with different credentials.

5. Registration: A registration phase is required to register the user's, the devices's or the server's information.

6. Offline phase: Offline or pre-deployment phase is required to setup the network or for updates.

Table 2.2: Classification of Authentication Techniques based on Characteristics of Authentication.

	[11]	[7]	[5]	[6]	[10]	[14]	[4]	[9]	[13]	[28]	[12]	[15]	proposed
Mutual authentication	✓			✓	✓					✓			✓
Additional Hardware			✓		✓	✓		✓		✓	✓		
Multiple Credentials		✓	✓	✓		✓	✓		✓	✓			✓
Multiple Authentication Levels			✓						✓			✓	
Registration				✓			✓	✓		✓	✓		✓
Offline Phase	✓				✓	✓		✓					

2.5 Comparison of Authentication Techniques

In order to compare the authentication techniques proposed in literature, they can be measured, firstly, on the basis of the evaluation model used and secondly, based on their resistance to some attacks.

Evaluation Model The evaluation model used can be one or a combination of the following:

1. Experimental results by implementation using a particular programming language, a prototype, or testbed.
2. Experimental results by simulation using a particular network simulator.
3. Theoretical evaluation using formal security analysis for the proposed technique.
4. Performance analysis by analyzing the required CPU time, number of steps, memory space, message size, and/or probability of breaking the technique [29].

Table 2.3 records the evaluation models used by the authors who proposed the authentication techniques mentioned above.

Table 2.3: Comparison of Techniques based on Evaluation Model Used.

	[11]	[7]	[5]	[6]	[10]	[4]	[9]	[13]	[28]	[15]	proposed
Implementation			✓			✓	✓				
Simulation	✓				✓	✓			✓	✓	✓
Theoretical Evaluation	✓	✓		✓	✓	✓	✓	✓	✓		✓
Performance Analysis	✓	✓			✓	✓	✓			✓	

Security Attacks The following security attacks have been considered in the proposals and for comparison:

1. Man-in-the-Middle Attack: An attacker inserts him/herself into a conversation between two parties, impersonates one or both parties and gains access to information that the two parties were trying to send to each other. This attack

takes advantage of the weakness in the authentication process and alters the communication between the communicating entities. This attack can be performed if the platform uses an inadequate way to authenticate parties in the network.

2. Timing Attack: Timing attacks enable a malicious actor to extract secrets maintained in a security system by observing the time it takes the system to respond to various queries and in turn figuring out how the encryption algorithm is implemented [30].

3. Insider Attack: This attack happens when an authorized member node communicates with and attacks other members of the network [31].

4. Impersonation Attack: The malicious attacker uses a trusted user's identity and authentication credentials to get access and send malicious messages to other entities [31].

5. Replay Attack: A malicious node may repeat the data continuously which will cause irregular and malicious behaviour in the network [25]. A smart home could be susceptible to a replay attack if an inadequate methodology is used to authenticate parties or packets.

6. Password Change/ Guessing Attack: This is the most damaging threat where an attcker takes advantage of the vulnerability in the security scheme and successfully changes/guesses the password of a registered entity [32].

7. Forgery Attack: An adversary can masquerade as a legal user and get access to the necessary credentials [32].

8. Eavesdropping Attack: This is a passive attack, which occurrs in the mobile ad-hoc network. The attacker finds out some secret or confidential information by listening to the communication between the entities [25].

9. Denial of Service (DoS) Attack: In denial of services attack, malicious node keeps sending messages to the receiver node and consumes the bandwidth of the network. This affects services availability and trusted networks will have to wait for their messages to be delivered to the receiver [25]. In short, the attacker overloads the system with too many requests causing it to crash eventually.

10. Stolen Smartcard Attack: An adversary knows the information in the user's smart card and can use it to successfully complete authentication [32].

11. Rainbow table attack: A rainbow table is a precomputed lookup table that stores password hashes. In this attack, a perpetrator tries to use a rainbow hash table to recover a password based on its hash value [33].

12. Brute-force attack: An attacker performs an exhaustive search where he or she uses trial and error method to explore all possible passwords of the user [33].

Table 2.4 enumerates the security attacks used for comparison and those considered by the authors while developing the authentication technique proposed.

2.6 Chapter Summary

This chapter listed the metrics to evaluate an IoT authentication scheme, classified and compared the authentication techniques proposed by other researchers in order to show the need for secure authentication in the IoT network.

Table 2.4: Comparison of Techniques based on Security Attacks.

	[4]	[5]	[6]	[7]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[28]	proposed
Man-in-the-Middle	✓	✓	✓		✓	✓	✓	✓	✓			✓	✓*
Timing				✓									
Insider		✓											
Impersonation			✓		✓				✓			✓	
Replay	✓		✓		✓	✓	✓		✓			✓	✓
Password Changing/ Guessing		✓	✓		✓					✓			✓
Forgery				✓	✓		✓	✓					✓
Eavesdropping				✓		✓	✓						
Denial-of-Service (DoS)											✓		✓
Stolen Smart-card		✓	✓										✓
Rainbow Table		✓	✓										
Brute-Force		✓	✓										

*subject to specific considerations

Chapter 3

Proposed Security Scheme Overview

This section explores the different phases in the lifecycle of a device in the IoT network from the manufacturing phase to the functioning phase.
Communication between two different entities in the IoT have a number of phases:

1. Authentication phase: During which the entities securely establish each others' identities.

2. Key exchange phase: Entities agree on a symmetric session key to encrypt the data exchanged.

3. Data transfer phase: Application data is encrypted using the agreed keys and exchanged between the two entities.

In the protocol proposed, we explore two general problems in cryptography in the IoT:

1. Entity Authentication: An entity ensures that one or more entities are, in fact, who they say they are.

2. Key-Exchange: Two or more entities agree on a common key or keys that can be used for secure communication.

We will be focussing on the design and develpoment of an authentication protocol for device-to-device authentication without human intervention. We will also be talking about the pre-deployment,deployment and functioning phases.

3.1 Notations

The abstract notations listed in Table 3.1 will be used to represent the data exchanged in the security protocols and the cryptographic processes.

Table 3.1: Notations.

Notation	Description
D_1	IoT Device 1 (e.g. a TV set)
D_2	IoT Device 2 (e.g. a microwave)
P_1	One-time Access Password of Device D_1
G	Gateway
C	Controller
U_D	User's Personal Device (e.g. a phone)
ID_1, ID_2	Identity of Device 1 and Device 2
K_{u_1}, K_{u_2}	Public Key of Device 1 and Device 2
K'_{u_1}, K'_{u_2}	Private Key of Device 1 and Device 2
ID_g	Identity of Gateway
K_{uG}, K_{uC}	Public key of Gateway and Controller, respectively
IP_{v6}	Virtual Identity
T_c	Current Timestamp
N_1, N_2, N_3, N_4	Nonces: an unpredictable bit string
H, H_1	Hashes: ensure that transmitted messages have not been tampered with
H(IPv6)	Hash of IPv6 address of Device 1
K_{12}	Shared key between Device 1 and Device 2
$M -> N$	Message sent from M to N
M'	New value of M
$En\{M\}$	Encrypted value of a message M

3.2 Pre-Deployment Phase

The pre-deployment phase materializes at the manufacturer's site before the devices are shipped to the trader or user. The manufacturer will load the devices with the information and cryptographic details regarding the cloud server. This will help the creation of a trusted communication bridge between the manufacturer and the devices. This will be useful to the manufacturer to execute procedures like software update or download new firmware onto the devices without the necessity of a physical link.

The controller in the remote server generates the device identity, ID_1 and using a Random Number Generator, it computes the device's private key and public key.

It also generates a one-time access password. These keys - public and private key of device D_1, the device's identification value and access password are loaded into the device's memory over a secure channel, i.e . one that is resistant to tampering and overhearing, achieved through the cryptographic protocols implemented by the manufacturer. The controller in the remote server saves these in its database and also sends the device ID, ID_1, and the one-time access password, P_1 to the device D_1 as shown:

$$C \rightarrow D_1 : En\{ID_1, P_1, K_{u1}, K'_{u1}\} \tag{3.1}$$

where $En\{\ \}$ implies encryption of the data within the braces according to Table 2.4.

3.3 Deployment Phase

When the user gets the IoT device, D_1, the user requests authentication information from the cloud server so that he or she can access the device. The cloud server's controller sends the user the one-time access password, which when entered into the IoT D_1 wil grant the user access to it and its services.

Once the user gains access to IoT D_1, the device can connect to the home network server through the smart gateway, G. The user's personal device sends the identity of the smart gateway, ID_G, and its public key, K_{u_G}, to D_1:

$$U_D \rightarrow D_1 : ID_G, K_{u_G} \tag{3.2}$$

The device, D_1, encrypts and sends to the gateway its own ID along with the current timestamp to the smart gateway to be registered in the home network. On receiving the data from D_1, the gateway will decrypt the received message and check the timestamp. If valid, it will compute $H_1(ID_G, T_c))$ which was the encryption key used to encrypt the ID of D_1. This step is shown in the following equation:

$$D_1 \rightarrow G : \{T_c\}K_{u_G}, \{ID_1\}H_1(ID_G, T_c) \tag{3.3}$$

where T_c is the current timestamp.

Once the gateway obtains the device identity, it forwards to the controller in the remote server - the device identity, the current timestamp and a nonce N_1, all encrypted appropriately as shown in the equation below:

$$G \rightarrow C : \{T_c, \{ID_1, N_1\}K'_{u_c}\}K_{u_C} \tag{3.4}$$

The timestamps convey information about when an event occurred which makes this value important. The nonce is a value that is used only once and is used to distinguish one instance of the protocol from the other. The timestamps and nonces are chosen in such a way as to prevent replay attacks. The controller in the cloud server checks the received timestamp with the current timestamp. If valid, it'll obtain N_1, ensure N_1 has not been received before and then obtain the device ID. Controller generates a virtual identity *IPv6* for D_1, stores it in its database with the corresponding ID, ID_1. It sends to the gateway concatenated values of the hash of *IPv6*, the gateways public key encrypted by the device's private key concatenated with the current timestamp, encryting this part of the message by the gateway's public key. This is depicted in the equation below:

$$C \rightarrow G : H(IPv6), \{\{K_{u_G}\}K'_{u_1}, T_c\}K_{u_G} \tag{3.5}$$

In the next step, G verifies the timestamp and stores the *H(IPv6)* with the corresponding device ID, ID_1 in its database. The gateway then forwards the *H(IPv6)* and the public key of the gateway encrypted by the device's private key as received from the controller and current timestamp, all encrypted by D'_1s public key. D_1 checks the received timestamp and if it is valid, it stores *H(IPv6)* in its memory. This step is shown by the below equation:

$$G \rightarrow D_1 : H(IPv6), \{\{K_{u_G}\}K'_{u_1}, T_c\}K_{u_1} \tag{3.6}$$

The smart gateway can communicate with other gateways and also has a local database to store all the information which is constantly accessed by the home network and the devices in it. The smart gateway updates the local information to the cloud periodically. The cloud has information about all the home networks and the devices connected in them whereas the smart gateway only has information of the devices connected in its home network.

3.4 Functioning Phase

During the normal operation of the home network and the devices in this network, the user uses a personal device to request D_1 to perform a particular operation. For this, the device D_1 has to communicate with another device in the home network, D_2. D_1 initially forwards this communication request to the gateway.

$$D_1 \rightarrow G : H(IPv6), \{T_c, D_2\}K_{u_G} \tag{3.7}$$

The gateway checks the timestamp and if valid, it checks in database for information about D_2 and sends to D_1 - K_{12}, which is a shared key that can be used for communication between D_1 and D_2, concatenated values of a nonce N_2 and the current timestamp and K_{u_G} concatenated with K_{12} encrypted by the device D_2's private key encrypted by the device's public key as below:

$$G \rightarrow D_1 : \{\{K_{u_G}, K_{12}\}K'_{u_2}\}, T_c, N_2, K_{12}\}K_{u_1} \tag{3.8}$$

$\{K_{u_G}, K_{12}\}K'_{u_2}$ sent to D_1 by G will be forwarded as it is to D_2 and this assures D_2 that the key was indeed shared by its gateway. D_1 checks the timestamp and if it is valid, it obtains the shared key K_{12}. D_1 forwards $\{K_{u_G}, K_{12}\}K'_{u_2}$ to D_2 along with a nonce N_3 and the current timestamp concatenated and encrypted as shownbelow:

$$D_1 \rightarrow D_2 : \{\{K_{u_G}, K_{12}\}K'_{u_2}, \{N_3\}K_{12}, H1(K_{12}, T_c)\}K_{u_2} \tag{3.9}$$

D_2 decrypts the received message and obtains K_{12} from $\{K_{u_G}, K_{12}\}K'_{u_2}$ assured that it has been provided by its gateway. Then it decrypts and obtains N_3 using K_{12}. D_2 knows the hash function and K_{12} and can finally obtain T_c to validate the timestamp.

D_2 replies to D_1 with another nonce N_4 concatenated with the received nonce N_3. When D_1 receives N_3 and a valid timestamp encrypted by the shared key K_{12}, D_2 is authentic to D_1. In order to authenticate itself to D_2, D_1 replies with N_4 encrypted by the shared key which is known only by D_1 and D_2. Both these steps are illustrated in the equations below:

$$D_2 \rightarrow D_1 : \{\{N_3, N_4\}K_{u_1}, T_c\}K_{12} \tag{3.10}$$

$$D_1 \rightarrow D_2 : \{N_4\}K_{12} \tag{3.11}$$

3.5 Chapter Summary

In this chapter, the processes the device goes through from the manufacturing phase to the functioning phase and how it is registered with the cloud are discussed. The design and implementation of this is explained in the next section.

Chapter 4

Protocol Design and Simulationl

This chapter focuses on explaining how the authentication protocol proposed in Chapter 3 has been implemented in AVISPA and analyzing the security properties it verifies. We introduce the design of the proposed protocol and its implementation in HLPSL. We demonstrate the simulation of the protocol and also the verification of the protocol against attacks using the back-ends of the AVISPA tool.

4.1 Formal Verification Tool -AVISPA

The proposed security protocol is designed, simulated and tested using AVISPA, an acronym for Automated Validation of Internet Security Protocol and Applications. AVISPA is a push-button tool for the automated validation of Internet security-sensitive protocols and applications. It is a role-oriented language in which each participating agent plays a different role during the protocol execution and each role is independent of the other. It specifies protocols and their security properties by means of a modular and expressive formal language. It integrates different back-ends implementing a variety of automatic analysis techniques for protocol falsification by finding an attack on the input protocol, and abstraction-based verification methods both for finite and infinite numbers of sessions. This tool uses a Dolev Yao attacker model, giving the attacker full control over the network. AVISPA verifies the confidentiality, integrity, authentication of communication and of data origin, anonymity, non-repudiation and key-management properties of a given protocol or communication exchange. This tool is a web-based, platform-independent realization and is compatible with the common operating systems.

The intruder is modeled using the Dolev-Yao's threat model, which represents the most powerful attacker possible in a network. This model is the most widely accepted model for analyzing security protocols. It is assumed that if a protocol is shown to be secure against this threat model, it is secure against less powerful variations of attacks or other real-world attacks. This threat model is distinguished from other models by several capabilities:

1. The secret information is assumed to be indivisible and the attacker does not exploit partial messages. The message is considered to be a secret unless it is compromised in its entirety.

2. The intruder can play the role(s) of regular principals or agents in the protocol.

3. The attacker can forward the messages even if he/ she cannot read it.

4. No information about a message can be retrieved from its encrypted form or hash-code without the required key.

5. The attacker can not only read but also delete the messages and also forge new messages.

6. The attacker can be an internal agent to the protocol and engage in a protocol run to learn information which can be used later in another protocol run.

In general, it can be stated that this threat model has complete control over the network but since the cryptographic primitives are assumed to be perfect, it is not able to perform cryptanalysis.

The architecture of the AVISPA Tool is depicted in Fig. 4.1. A user interacts with the tool by specifying a security problem - a protocol paired with a security property that the protocol is expected to achieve, in the High-Level Protocol Specification Language (HLPSL). The HLPSL is an expressive, modular, role-based, formal language for modeling communication and security protocols. The HLPSL features allow one to specify protocols without resorting to specific techniques to simplify the protocols first. HLPSL specifications are automatically translated into the IF by the HLPSL2IF translator. The Intermediate Format (IF) is a lower level language at an accordingly lower abstraction level. These translations in turn, serve as input to the various back-ends that are analysis tools of the AVISPA tool-set.

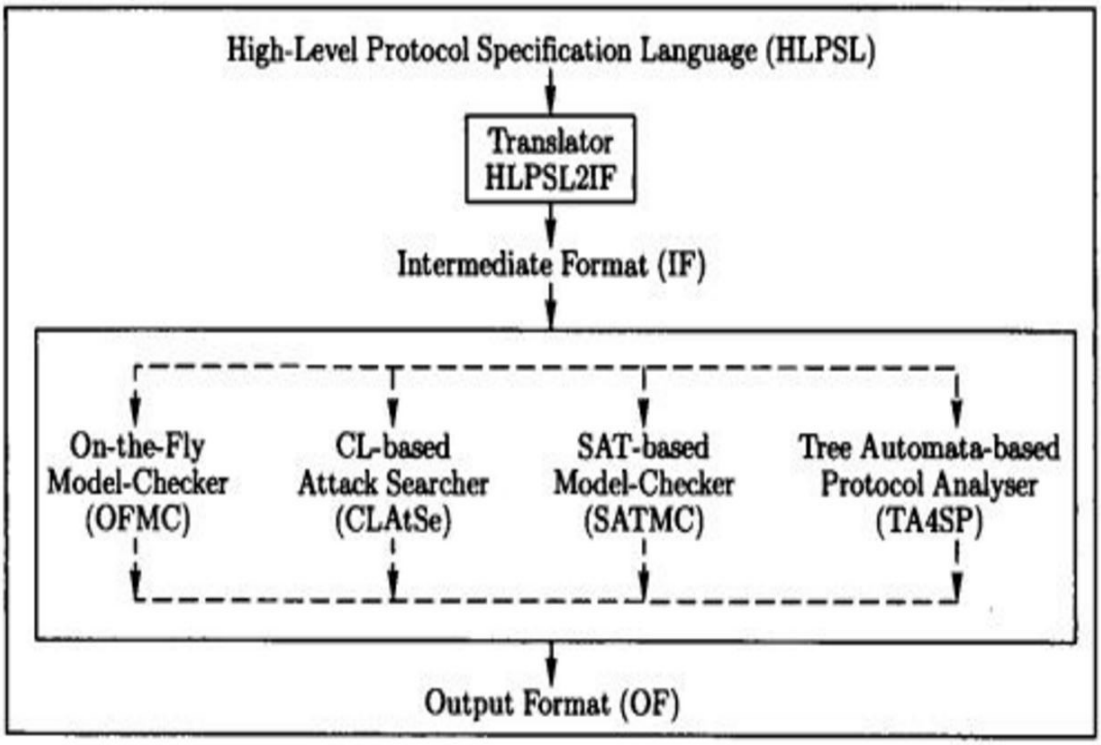


Figure 4.1: Architecture of the AVISPA.

The Back-Ends are used to provide protocol falsification, bounded and unbounded verification. AVISPA has the following four back-end tools for mathematical processing.

1. OFMC: The On-the-fly Model-Checker (OFMC) is used for a protocol where the algebraic properties of the cryptographic functions are important. This can be employed not only for efficient falsification of protocols i.e. fast detection of attacks, but also for verification i.e. proving the protocol correct for a bounded number of sessions, without bounding the messages an intruder can generate.

2. CL-AtSe: Constraint-Logic-based Attack Searcher (CL-AtSe) is a constraint based system that strategizes to translate the security protocol specification into a set of constraints which can be effectively used to find attacks on protocols.

3. SATMC: The SAT-based Model-Checker (SATMC) takes a transitional state

from the IF and builds a propositional formula representing a violation of the security properties. The propositional formula is then fed to a state-of-the-art SAT solver and any model found is translated back into an attack.

4. TA4SP: Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP) tool points out the vulnerability of a protocol or predicts its soundness by a careful estimation of the intruder's capabilities. It computes either an over-approximation or an under-approximation of the intruder knowledge which shows that the protocol is flawed for a given secrecy property.

4.2 Modeling of the protocol in AVISPA

The authentication protocol for the IoT is the outcome of this thesis, and is modeled using AVISPA tool for verifying its security properties. The messages involved in the cryptography and security of the authentication process are taken into consideration. The protocol is modeled using HLPSL specification. The ultimate goal of the protocol is mutual authentication be established between the devices in the IoT network.

4.2.1 HLPSL Syntax

The abstract notations listed in Table 4.1 will be used to represent the data exchanged in AVISPA.

Table 4.1: AVISPA notations.

Notation	Description
$D_1 -> D_2 : M$	Message M sent from D_1 to D_2
$action1/\ action2$	This operator indicates in HLPSL that action1 and action2 are operated simultaneously
M_k	Message M encrypted with key k
T_c'	New value for the current timestamp T_c
M'	New value of any message M
$State = 1 = \| > State = 2$	This operator indicates in HLPSL that the state transitions from a value of 1 to a new value 2.

The entities involved in the communication process are modeled as roles with their message exchanges. Apart from the initiator and the receiver, the environment and the session of the protocol are also roles in HLPSL. The roles can be basic or

composed depending on if they are constituent of one agent or more. A session is created between the roles to achieve the required message exchanges. The session also considers an intruder in the process. This is modeled using one session which includes the intruder, having a knowledge of the keys of the sender and the receiver, and the identities of the devices involved. Finally the environment is created for all the sessions simultaneously. This is explained in detail:

1. Basic Roles - It is easy to translate a protocol into HLPSL if it is written in Alice-Bob (A-B) notation. A-B notation for particular protocol is:

   A -> S: {K12}Kas
   S -> B: {K12}Kbs

   Here, A and B are agents, S is the server, Kas is the shared key between A and S, and Kbs is the shared key between B and S. In this protocol, A wants to set up a secure session with B by exchanging a new session key, K12 with the help of a trusted server. A generates the session key, K12 which is intended to be shared with B. A encrypts this key with Kas and send it to S. Then S decrypts message, re-encrypts K12 with Kbs and sends it to B. After this exchange A and B share the new session key and can use it to communicate with one another. Here is an example of a basic role declaration extracted from the HLPSL specification of this protocol:

```
role role_A(A:agent,B:agent,Kas, Kbs:symmetric_key,
                SND,RCV:channel(dy))
played_by A
def=
        local
                State:nat,K12:symmetric_key
        init
                State := 0
        transition
                1. State=0 /\ RCV(start)
                        =|> State':=1
                        /\ SND(K12)_Kas
end role
```

2. Transitions - System behavior in HLPSL is modeled as a 'State'. Every state has variables which are responsible for the state transitions, i.e, when a variable changes, the state takes a new form. The communicating entities are called roles and they own the variables. The parties involved in our protocol are Device 1 - D_1, Device 2 - D_2, the Gateway - G and the Controller - C.

   The transition part contains a set of transitions. Each represents the receipt of message and the sending of a reply message. The example of simple transition is as follows:

   ```
   Step 1: State = 0 /\ RCV({K12}_Kas) =|> State :=1
   ```

   Here, Step 1 is the name of the transition. Step 1 specifies that if the value of State is equal to zero and a message, which contains some value K12 encrypted with Kas is received on a channel RCV, then a transition fires which sets the new value of State to 2 and sends the same value K12 encrypted with Kbs on a channel SND.

3. Composed Roles - The basic roles of A and B are composed together in sessions where the knowledge shared between the roles (public keys for instance) are made explicit. Composed roles contains one or more basic roles and execute in parallel. It has no transition section. The operator in $role_A / role_B$ indicates that the roles should execute in parallel. Here, the type declaration channel (dy)stands for the Dolev-Yao intruder model. According to this model, the intruder has full control over the network, such that all the messages sent by the agents will go to the intruder as well. All the agents can send and receive on whichever channel they want. Here is an example of a session extracted from the HLPSL specification of the protocol specified above:

   ```
   Role session(A,B,S : agent, Kas, Kbs : symmetric key)
   def=
   Local SND3,RCV3,SND2,RCV2,SND1,RCV1 :channel (dy)
   Composition
            role_A (A,B,S, Kas, SND1,RCV1)
            /\ role_B (B, A, S, Kbs, SND2, RCV2)
            /\ role_S (S, A, B, Kas, Kbs, SND3, RCV3)
   end role
   ```

4. Environment - Finally, the environment used for protocol execution is defined, where 'i'denotes the intruder and 'inv(ki)'denotes the intruder's private key. The environment also defines the initial knowledge of the intruder and the initial setting for the sessions, i.e. how many session runs are executed and who runs these sessions.

```
role environment() def=
const a, b, s : agent,
kas,kbs : symmetric_key
intruder_knowledge = {a, b, ki, inv(ki)}
composition
session(a,s,kas) /\ session(b,s,kbs) /\ session(a,i,ki)
end role
```

4.2.2 Protocol Modelling in HLPSL

We implement a simple smart home IoT network consisting of the home devices, the gateway and the controller in the remote server. Initially when device, D_1, joins the network, it needs to authenticate itself to the gateway, G which is shown as below:

$$D_1 \rightarrow G : \{Tc\}K_{u_G}, \{ID_1\}H_1(ID_G, T_c) \tag{4.1}$$

This can be modeled in HLPSL as:

```
State=0 /\ RCV(start)
        =|> State':=1
        /\ Tc':=new()
        /\ SND({Tc'}_Kug.{IDd1}_H1(IDg.Tc'))
```

On receiving this, if the timestamp is valid, the gateway will compute the $H_1(ID_G, T_c)$, obtain the device's ID and forward the information in a digitally encrypted format to the controller. We illustrate the A-B notation of this step:

$$G \rightarrow C : \{T_c, \{ID_1, N_1\}K'_{u_c}\}K_{u_C} \tag{4.2}$$

This can be modeled in HLPSL as:

```
State=0 /\ RCV({Tc'}_Kug.{IDd1}_H1(IDg.Tc'))
        =|> State':=1
```

```
/\ N1':=new()
/\ secret(N1',sec_1,{})
/\ SND({Tc'.{IDd1.N1'}_inv(Kuc)}_Kuc)
```

The controller checks for the validity of the received timestamp and ensures that the nonce N_1 has not been received before. The controller generates and stores the IP_v6 in the cloud memory with the corresponding device ID and also sends the same in encrpted format to the gateway. The encrypted messages sent from the controller to the gateway in A-B notation:

$$C \rightarrow G : H(IP_v6), \{\{K_{u_G}\}K'_{u_1}, T_c\}K_{u_G} \tag{4.3}$$

The gateway decrypts the encrypted messages from the controller, checks the timestamp, stores the IP_v6 with the corresponding device ID in its database and re-encrypts this to forward it to the device, D_1 as below:

$$G \rightarrow D_1 : H(IP_v6), \{\{K_{u_G}\}K'_{u_1}, T_c\}K_{u_1} \tag{4.4}$$

This can be modeled in HLPSL as:

```
State=0 /\ RCV({Tc'.{IDd1.N1'}_inv(Kuc)}_Kuc)
        =|> State':=1
        /\ secret(N1',sec_1,{})
        /\ SND(H(IPv6).{{Kug}_inv(Kud1).Tc'}_Kug)

State=1 /\ RCV(H(IPv6').{{Kug}_inv(Kud1).Tc}_Kug)
        =|> State':=2
        /\ SND(H(IPv6').{{Kug}_inv(Kud1).Tc}_Kud1)
```

Here, after the device D_1 exchanges information with the gateway and the controller, the controller assigns, computes and sends a hash of the device's virtual identity, IPv6, to D_1 through the gateway. In this communication, nonces and the current timestamps are shared which help to prevent replay attacks.

Now, when the device D_1 wants to communicate with the device D_2, it connects with the gateway and asks for a session key to be used for D_1 to D_2 communication. The device sends the current timestamp and the identity of the device it wants to communicate with, concatenated together and encrypted by the gateway's public key, concatenated with the hash of its IP_v6. This is illustrated as:

$$D_1 \rightarrow G : H(IP_v6), \{T_c, D_2\}K_{u_G} \tag{4.5}$$

The gateway checks for a valid timestamp, searches in its database for the device D_2 and sends to D_1, a nonce, the shared secret session key, the current timestamp in addition to a message that can be decrypted only by the device, D_2, which is $\{K_{u_G}$, $K_{12}\}K'_{u_2}$. This is to assure D_2 that this key was indeed shared by the gateway itself. All these are concatenated together and encrypted by the device, D_1 's public key. The A-B notation is illustrated as:

$$G \rightarrow D_1 : \{\{K_{u_G}, K_{12}\}K'_{u_2}\}, T_c, N_2, K_{12}\}K_{u_1} \tag{4.6}$$

In the gateway's role, the gateway receiving the messages sent by device, D_1 and sending the encrypted secret key with the other values as mentioned above is modeled in HLPSL:

```
State=2 /\ RCV(H(IPv6).{Tc.D2}_Kug)
        =|> State':=3
        /\ K12':=new()
        /\ secret(K12',sec_2,{})
        /\ N2':=new()
        /\ SND({{Kug}_inv(Kud2).Tc.N2'.K12'}_Kud1)
```

D_1 decrypts the message from G, checks the current timestamp and the nonce to verify the message is not being replayed and the secret session key to be shared between D_1 and D_2. It concatenates the current timestamp, the message that can be decrypted only by D_2, a new nonce encrypted by the secret session key and a hash of the secret session key, all encrypted by the device D_2's public key and forwards the messages to D_2 illustrated below:

$$D_1 \rightarrow D_2 : \{\{K_{u_G}, K_{12}\}K'_{u_2}, \{N_3\}K_{12}, H1(K_{12}, T_c)\}K_{u_2} \tag{4.7}$$

D_1 receiving the messages sent by G and sending the above message to D_2 can be modeled in HLPSL:

```
State=2 /\ RCV({{Kug}_inv(Kud2).Tc.N2'.K12'}_Kud1)
        =|> State':=3
         /\ secret(K12',sec_2,{})
        /\ N3':=new()
```

```
/\ SND({{Kug}_inv(Kud2).{N3'}_K12'.H1(K12'.Tc)}_Kud2)
```

D_2 decrypts the received message, obtains K_{12} from $\{K_{u_G}, K_{12}\}K'_{u_2}$ assured that it is approved by its gateway and also checks the received timestamp against the current timestamp. It compares the key it got from this with the key from $H1(K_{12}$ to confirm that the agent thats communicating with it has the same key. D_2 sends to D_1, a concatenation of the nonces N_3 and N_4 encrypted by device $D-1$'s public key concatenated with the current timestamp. This message is encrypted by the shared secret session key K_{12} and the contents of this is used to authenticate itself to D_1. This is illustrated in the equation below:

$$D_2 \rightarrow D_1 : \{\{N_3, N_4\}K_{u_1}, T_c\}K_{12} \tag{4.8}$$

This process where D_2 receives the information from D_1, decrypts the message and sends the above mentioned values to D_1 is modeled in HLPSL as:

```
State=0 /\ RCV({{Kug}_inv(Kud2).{N3'}_K12'.H1(K12'.Tc')}_Kud2)
        =|> State':=1
        /\ secret(K12',sec_2,{})
        /\ N4':=new()
        /\ SND({{N3'.N4'}_Kud1.Tc'}_K12')
```

D_1 decrypts the message since it has the shared secret key, verifies the timestamp and obtains the nonces. Since it also received the nonce N_3, the device D_1 confirms the message was sent by device D_2 itself. Device D_1 replies with the received nonce N_4 encrypted by the shared secret session key to authenticate itself to D_2 as illustrated in the equation:

$$D_1 \rightarrow D_2 : \{N_4\}K_{12} \tag{4.9}$$

This step is modeled in HLPSL as:

```
SND({N4'}_K12)
```

In the main program this is included as:

```
State=3 /\ RCV({{N3.N4'}_Kud1.Tc}_K12)
        =|> State':=4
        /\ secret(K12',sec_2,{})
         /\ witness(D1,D2,auth_4,N4')
```

/\ SND({N4'}_K12)

In this section, we have discussed the processes leading to mutual authentication be established between the devices in the IoT network.

4.3 Protocol Simulation and Formal Analysis of Security Properties

AVISPA allows one to simulate a protocol that is designed in it. The simulation result of our proposed scheme is based on the two widely-accepted back-ends such as OFMC and CL-AtSe using the AVISPA web tool. Once the AVISPA model is developed using HLPSL, we use a security protocol animator (SPAN) to symbolically execute the HLPSL protocl specification. SPAN gives a better understanding of the protocol and confirms if the specification is executable. No intruder role has been introduced during the protocol simulation. SPANs protocol simulation builds Message Sequence Charts (MSC) corresponding to our HLPSLspecification and is shown in Fig.4.2:

The AVISPA tool verifies if the proposed protocol satisfies the security properties like authentication, integrity and secrecy. If the protocol is unsafe, it also gives the trace of the attack found. This feature of finding attacks and giving the attack trace makes AVISPA different from other tools.

4.3.1 Authentication

Authentication is the process of validating a device's identity by the other devices in communication, ensuring the reliability of origin of communication and guaranteeing that they are communicating with the desirable party. Mutual authentication is established in the proposed protocol through the shared session key generated by the gateway which is kept secret between the two devices it was intended for, D_1 and D_2. D_1 and D_2 authenticate themselves to each other by exchanging the nonces N_3 and N_4 respectively. Using the secret session key to encrypt their messages ensures that the data cannot be tampered by an intermediary agent.

4.3.2 Secrecy

The property of secrecy ensures that the information cannot be deciphered or is not disclosed to an unauthorized entity. The protocol accomplishes the goal of keeping the

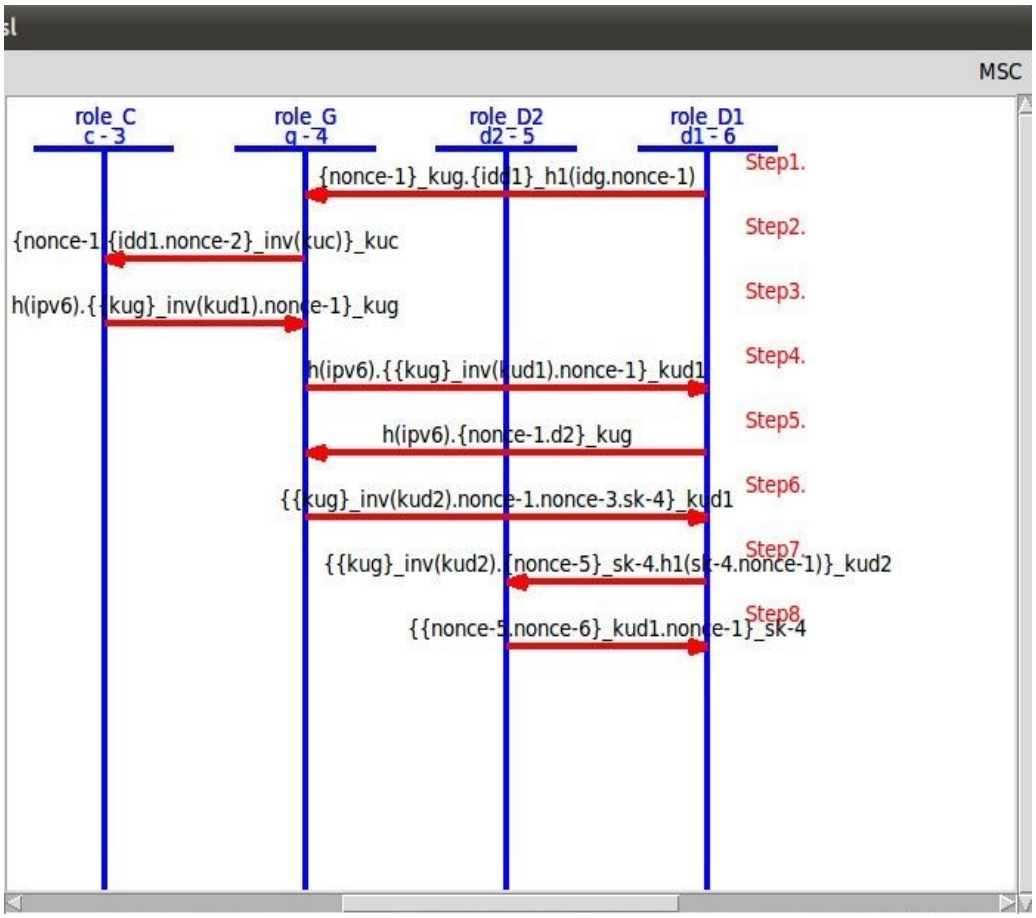


Figure 4.2: Protocol Simulation in AVISPA.

nonces N_1, N_3 and N_4 as a secret during the communication between the different entities. The protocol ensures this by encrypting the nonces such that only the intended receiver can decrypt the message. The shared key, K_{12}, is also ensured to be kept secret between the communicating parties.

4.3.3 Integrity

Integrity implies that the information has not been altered or distroyed in an unauthorized manner by an attacker/ entity/ process during the communication between the authorized entities. In addition to using the session key to ensure integrity, the hash of the data is also generated by the sender. When the receiver is able to verify that the received hash has been created with the same algorithm, it confirms that data integrity is not violated.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/hlpslGenFile.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.02s
  visitedNodes: 11 nodes
  depth: 10 plies
```

Figure 4.3: Protocol Verification with OFMC Back-End .

4.4 Simulation and Verification Results

The simulation consists of modeling of the protocol using HLPSL in AVISPA and building MSC's using SPAN. Now, we introduce an active intruder and verify the model using the back-ends provided in the tool. The verification results using OFCM back-end and CL-AtSE back-end are as produced in Fig.4.3 and Fig.4.4 respectively. The other two, SATMC and TA4SP, have reported NOT SUPPORTED and produced INCONCLUSIVE results.

The figures Fig.4.3 and Fig.4.4 confirm that the proposed protocol is SAFE under the two back-ends. AVISPA simulation results ensure that the proposed scheme is secure against active and passive attacks including replay and man-in-the-middle attacks. From this, it can be concluded that the protocol is safe. It indicates there is no authentication attack and secrecy attack on the protocol. The authentication of the participating parties is done by exchanging the corresponding nonces generated by them. There is no attack found on the session key by the intruder. Also the secrecy of the session key and the transferred messages between the parties are maintained. Including the hash function in the modeling of the protocol maintained the integrity also. The backend itself is the attack searcher that gives no information about any

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  /home/span/span/testsuite/results/hlpslGenFile.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed    : 22 states
  Reachable   : 8 states
  Translation: 0.01 seconds
  Computation: 0.00 seconds
```

Figure 4.4: Protocol Verification with CL-AtSE Back-End .

possible attack, indicating protocol is safe. Thus, the protocol is modeled and the security properties are verified using AVISPA tool.

4.5 Informal Security Analysis

In this section, we informally analyse the security of our proposed authentication scheme to show that our protocol provides strong security against the intruder attacks. In order to detect attacks like replay, parallel sessions, and DoS attacks, we used CL-AtSe backend which detects possible attacks and gives a trace of the attacks. As seen in the previous section, CL-AtSe backend confirmed that our protocol is safe from these attacks.

1. Replay Attack - Timestamps are generally used to resist replay attack. In some cases, this method may be affected by clock synchronization problem at any

point of time. The proposed protocol makes use of random nonces in addition to the timestamps to ensure the freshness of the message.

2. Password Guessing Attack - The OTP is generated at the manufacturer's end using a random password generation mechanism. Since the password is not dependent on any credentials that the attacker can obtain from the protocol, the probability of guessing is so negligible that the attacker fails to guess the correct parameters/ password.

4.6 Chapter Summary

In this chapter, the design for the proposed device-to-device authentication protocol is developed in HLPSL code. The protocol was verified using the AVISPA tool in which a flaw or error can be detected, verified and traced. A detailed protocol model for implementing the authentication was discussed. The back-ends of AVISPA show that the goals of secrecy and authentication hold true for the protocol and the results obtained were also illustrated. The next section explores the evaluation of the protocol.

Chapter 5

Theoretical Evaluation

This chapter explores the security of the protocol against attacks such as multi-protocol attack and reflection attack in detail. We mathematically demonstrate how our protocol and the cryptography used is secure against such attacks.

The attacks demonstated against the proposed protocol are:

1. Multi-protocol attack: Multi-protocol attack is an attack against an authentication protocol using mesages generated from a separate protocol to mislead some participants into successfully completing the protocol and exchanging data.

2. Reflection Attack: Reflection attack is an attack against a challenge-response authentication protocol where an intruder fraudulently authenticates itself to the originator of the information, ultimately hiding the attacker's own identity.

The proposed protocol authenticates communication between D_1 and D_2 and is as follows:

Step 1 $D_1 \rightarrow D_2 : \{\{K_{u_G}, K_{12}\}K'_{u_2}, \{N_3\}K_{12}, H1(K_{12}, T_c)\}K_{u_2}$

Step 2 $D_2 \rightarrow D_1$: $\{\{N_3,N_4\}K_{u_1}, T_c\}K_{12}$

Step 3 $D_1 \rightarrow D_2$: $\{N_4\}K_{12}$

5.1 Multi-protocol Attack

Consider the protocol below similar to the tailored protocol in [34], which has shown to successfully attack the modified Needham Schroeder Public-Key protocol:

Message 1 $D_2 \rightarrow D_1$: $\{\{M,N_4\}K_{u_1}, T_c\}K_{12}$

Message 2 $D_1 \rightarrow D_2$: $\{N_4, D_2\}K'_{u_1}$

The Message 1 in the tailored protocol is exactly the same format as Step 1 in our proposed protocol. The attacker resorts to interleaving messages from multiple protocols in order to attempt attacking the authentication protocol. This has been demonstrated below:

Step 1 $i_{D_1} \rightarrow D_2 : \{\{K_{u_G}, K_{12}\}K'_{u_2}, \{N_3\}K_{12}, H1(K_{12}, T_c)\}K_{u_2}$

Step 2 $D_2 \rightarrow i_{D_1}$: $\{\{N_3,N_4\}K_{u_1}, T_c\}K_{12}$

Message 2 $i_{D_2} \rightarrow D_1$: $\{\{M,N_4\}K_{u_1}, T_c\}K_{12}$

Message 2 $D_1 \rightarrow i_{D_2}$: $\{N_4, D_2\}K'_{u_1}$

The attack proceeds as follows:

1. The intruder, i_{D_1}, masquerading as D_1, sends message in Step 1 to D_2. The tailored protocol requires N_3 and M to have the same format.

2. The intruder will receive the response message in Step 2 from D_2.

3. The intruder, masquerading as D_2, forwards this message as Message 1 to D_1.

4. D_1 responds with Message 2 of the tailored protocol, which includes a publicly readable copy of the nonce N_4.

5. The intruder grabs this value of N_4 to send it to D_2 to authenticate itself.

This attack fails because although the intruder is able to force D_1 to decode the secret field N_4 from Step 2 and send it back to the intruder in a format that the intruder can read, he cannot encrypt and send it D_2 in the format expected by D_2. This is because the message in Step 3 is encrypted by K_{12}, which is unknown to the intruder. Thus the intruder's attempts to fraudulently authenticate itself without alerting the host entity proved futile.

5.2 Reflection Attack

Consider a dishonest entity that attempts to impersonate D_1 to D_2. An example of reflection attack on a simple protocol is as below:

Step 1 $i_{D_1} \rightarrow D_2 : D_1,N_{D_1}$

Step 2 $D_2 \rightarrow i_{D_1}$: D_2, N_{D_2}, $\{N_{D_1}\}K_{12}$

At this stage, the intruder is stuck as he cannot encrypt N_{D_2} to send back to D_2. But the intruder can start a second instance of the protocol and get D_2 to encrypt N_{D_2}.

Step 3 $\quad i_{D_1} \rightarrow D_2 : D_1, N_{D_2}$

Step 4 $\quad D_2 \rightarrow i_{D_1}: D_2, N_{D_2}, \{N_{D_2}\}K_{12}$

The intruder abandons the second instance and resumes the first instance.

Step 5 $\quad i_{D_1} \rightarrow D_2 : \{N_{D_2}\}K_{12}$

Reflection attack is possible on this protocol because the initiator does not have to prove his identity and the receiver encrypts and decrypts any message it receives without confirming the identity of the originator.

The protocol proposed in this work prevents the reflection attack. The attack attempt proceeds as follows:

1. Step 1 $\quad i_{D_1} \rightarrow D_2 : \{\{K_{u_G}, K_{12}\}K'_{u_2}, \{N_3\}K_{12}, H1(K_{12}, T_c)\}K_{u_{D_2}}$
2. Step 2 $\quad D_2 \rightarrow i_{D_1}: \{\{N_3, N_4\}K_{u_1}, T_c\}K_{12}$

Even if the intruder tries to run a second instance of the proposed protocol, an attack doesn't happen for two reasons:

1. The intruder cannot get any information about the nonce N_4 from Step 2. This is because the message sent by D_2 in Step 2 is such that the nonces can be decrypted only by D_1.
2. The nature of the challenges produced by D_1 and D_2 are different, which is achieved by including different information in the ciphertext sent between the two entities. This ensures that blindly forwarding a message from another instance will not allow the intruder to authenticate itself.

5.3 Chapter summary

This chapter explored multi-protocol attack proposed by and reflection attack to demonstrate the security of our protocol against these attacks. This shows that the protocol design yields better and secure performance in real-world scenarios.

Chapter 6

Conclusions

In this thesis, we start with a comparative analysis of the protocols proposed and implemented by the other researchers in this field of the IoT. This comparative analysis showed us that many of the protocols do not give mutual authentication between devices the importance it deserves. Although mutual authentication between devices is one of the most significant factors to ensure secure data communication between the devices, it has not been explored to a large extent. Some of the proposed protocols require additional hardware also which may not be a feasible option in all situations. Some of the protocols discussed even ignore the necessity of using multiple authentication credentials to ensure security. This motivated us to consider designing a protocol for mutual duthentication in the smart home network.

This thesis presented a Device-to-Device mutual authentication algorithm which can be used by various devices in the IoT network and is streamlined to be used for the smart home network. This protocol ensures secure data communication between the communicating entities using asymmetric encryption methodology, i.e. through the use of public key and private key encryption and decryption. This encryption-decryption process is used to exchange data between the devices safely and in a secure manner. This methodology is used by the device to request the smart gateway for a secret session key that can be used for communication with another device. IoT Device 1 will send the key, sent by the gateway, to IoT Device 2. So the two devices agree on using the shared session key which is to be kept as a secret between them. This is possible only after the two devices have mutually authenticated themselves to each other. This session key is then used to encrypt and decrypt the communication betweeen these two IoT devices. The session key will be changed after every session and this ensures that the data exchange will be secure every single time.

This protocol is modeled in SPAN/ AVISPA tool. A brief explanation of the AVISPA tool is also given and the different back-ends that the tool employs is also described. The architecture of the AVISPA tool gives a better understanding of the process this security tool is involved in. The programming was done in HLPSL. Since it is easier to convert from A-B notation to HLPSL, the protocol was initially programmed in A-B notation. A protocol simulator also simulates the protocol exactly as it is programmed into the AVISPA tool. A formal verification of the protocol is performed usinf the back-ends OFMC and CL-AtSe. Both these back-ends, after careful verification, summarized the protocol to be SAFE. It needs to be noted that although the protocol is said to be safe by the AVISPA tool, there could be new attacks possible in the future which AVISPA may not have had the capability to check for. A theoretical evaluation was also done on the protocol. We explored this by using the Multi-protocol attack and the Reflection attack. The protocol was still shown to withstand these attacks. Thus it can be conluded that the protocol is safe in real-world scenarios.

Future work that can be done in this involves

1. Extending the protocol to consider cases where the IoT device leaves one particular home network and joins another network.

2. An IoT device in one home network communicating with an IoT device in another home network. Both these home networks may or may not use different communication protocols.

3. Extending the protocol to include access control so that devices have access to other devices such that this prevents any security breach from happening if one of the devices is compromised.

Appendix A

HLPSL Code

```
role role_D2(D1:agent,D2:agent,G:agent,C:agent,Kud1:public_key,
Kud2:public_key,Kug:public_key,Kuc:public_key,IDd2:text,
H:function,H1:function,SND,RCV:channel(dy))
played_by D2
def=
        local
                State:nat,N3:text,Tc:text,K12:symmetric_key,
                N4:text
        init
                State := 0
        transition
                7. State=0 /\ RCV({{Kug}_inv(Kud2).{N3'}_K12'.
                                H1(K12'.Tc')}_Kud2)
                                =|> State':=1
                                /\ secret(K12',sec_2,{})
                                /\ N4':=new()
                                /\ SND({{N3'.N4'}_Kud1.Tc'}_K12')
                9. State=1 /\ RCV({N4}_K12)
                                =|> State':=2
                                /\ secret(K12',sec_2,{})
end role

role role_D1(D1:agent,D2:agent,G:agent,C:agent,Kud1:public_key,
```

```
Kud2:public_key,Kug:public_key,Kuc:public_key,IDd1:text,
H:function,H1:function,IDg:text,SND,RCV:channel(dy))
played_by D1
def=
        local
                State:nat,IPv6:text,N2:text,N3:text,Tc:text,
        K12:symmetric_key,N4:text
        init
                State := 0
        transition
        1. State=0 /\ RCV(start)
                        =|> State':=1 /\ Tc':=new()
                        /\ SND({Tc'}_Kug.{IDd1}_H1(IDg.Tc'))
        4. State=1 /\ RCV(H(IPv6').{{Kug}_inv(Kud1).Tc}_Kud1)
                        =|> State':=2
                        /\ SND(H(IPv6').{Tc.D2}_Kug)
        6. State=2 /\ RCV({{Kug}_inv(Kud2).Tc.N2'.K12'}_Kud1)
                        =|> State':=3
                        /\ secret(K12',sec_2,{})
                        /\ N3':=new()
                        /\ SND({{Kug}_inv(Kud2).
                        {N3'}_K12'.H1(K12'.Tc)}_Kud2)
        8. State=3 /\ RCV({{N3.N4'}_Kud1.Tc}_K12)
                        =|> State':=4
                        /\ secret(K12',sec_2,{})
                        /\ witness(D1,D2,auth_4,N4')
                        /\ SND({N4'}_K12)
end role

role role_G(D1:agent,D2:agent,G:agent,C:agent,
Kud1:public_key,Kud2:public_key,Kug:public_key,
Kuc:public_key,IDg:text,IDd1:text,IDd2:text,
H:function,H1:function,SND,RCV:channel(dy))
played_by G
def=
```

```
        local
                State:nat,N1:text,IPv6:text,N2:text,Tc:text,
                K12:symmetric_key
        init
                State := 0
        transition
                1. State=0 /\ RCV({Tc'}_Kug.{IDd1}_H1(IDg.Tc'))
                                =|> State':=1 /\ N1':=new()
                                /\ secret(N1',sec_1,{})
                                /\ SND({Tc'.{IDd1.N1'}_inv(Kuc)}_Kuc)
                3. State=1 /\ RCV(H(IPv6').{{Kug}_inv(Kud1).Tc}_Kug)
                                =|> State':=2 /\ SND(H(IPv6').
                                {{Kug}_inv(Kud1).Tc}_Kud1)
                5. State=2 /\ RCV(H(IPv6).{Tc.D2}_Kug)
                                =|> State':=3 /\ K12':=new()
                                /\ secret(K12',sec_2,{})
                                /\ N2':=new()
                                /\ SND({{Kug}_inv(Kud2).Tc.
                                N2'.K12'}_Kud1)
end role

role role_C(D1:agent,D2:agent,G:agent,C:agent,
Kud1:public_key,Kud2:public_key,Kug:public_key,
Kuc:public_key,IDd1:text,IDd2:text,IDg:text,
H:function,IPv6:text,SND,RCV:channel(dy))
played_by C
def=
        local
                State:nat,N1:text,Tc:text
        init
                State := 0
        transition
                2. State=0 /\ RCV({Tc'.{IDd1.N1'}_inv(Kuc)}_Kuc)
                        =|> State':=1 /\ secret(N1',sec_1,{})
                        /\ SND(H(IPv6).{{Kug}_inv(Kud1).Tc'}_Kug)
```

```
end role

role session1(H1:function,D1:agent,D2:agent,G:agent,
C:agent,Kud1:public_key, Kud2:public_key,
Kug:public_key,Kuc:public_key,IDd1:text,IDd2:text,
IDg:text,H:function,IPv6:text)
def=
        local
                SND4,RCV4,SND3,RCV3,SND2,RCV2,SND1,
                RCV1:channel(dy)
        composition
role_C(D1,D2,G,C,Kud1,Kud2,Kug,Kuc,IDd1,IDd2,IDg,
        H,IPv6,SND4,RCV4)
/\ role_G(D1,D2,G,C,Kud1,Kud2,Kug,Kuc,IDg,IDd1,
        IDd2,H,H1,SND3,RCV3)
/\ role_D2(D1,D2,G,C,Kud1,Kud2,Kug,Kuc,IDd2,
        H,H1,SND2,RCV2)
/\ role_D1(D1,D2,G,C,Kud1,Kud2,Kug,Kuc,IDd1,
        H,H1,IDg,SND1,RCV1)
end role

role environment()
def=
        const
                hash_0:function,h:function,idd2:text,
                kuc:public_key,kud2:public_key,c:agent,
                d2:agent,h1:function,d1:agent,g:agent,
                kud1:public_key,kug:public_key,idd1:text,
                idg:text,ipv6:text,sec_1:protocol_id,
                sec_2:protocol_id,auth_3:protocol_id,
                auth_4:protocol_id
intruder_knowledge = {idd1,idd2,idg,kud1,kud2,kug,kuc}
        composition
                session1(h1,d1,d2,g,c,kud1,kud2,kug,kuc,idd1,
                idd2,idg,h,ipv6)
```

```
end role

goal
        secrecy_of sec_1
        secrecy_of sec_2
        authentication_on auth_3
        authentication_on auth_4
end goal

environment()
```

Bibliography

[1] K. Ashton, "That 'internet of things' thing http://www.rfidjournal.com/articles/view?4986,," *RFID Journal*, 2009.

[2] PricewaterhouseCoopers, "Sensing the future of the internet of things http://www.pwc.com/us/en/increasing-it-effectiveness/assets/future-of-the-internet-of-things.pdf," 2014.

[3] P. Madsen, "Authentication in the iot challenges and opportunities http://www.secureidnews.com/news-item/authentication-in-the-iot-challenges-and-opportunities/," 2015.

[4] V. L. Shivraj, M. A. Rajan, M. Singh, and P. Balamuralidhar, "One time password authentication scheme based on elliptic curves for internet of things (iot)," pp. 1–6, Feb 2015.

[5] M. A. Crossman and H. Liu, "Study of authentication with iot testbed," in *Technologies for Homeland Security (HST), 2015 IEEE International Symposium on*, pp. 1–7, April 2015.

[6] N. Shone, C. Dobbins, W. Hurst, and Q. Shi, "Digital memories based mobile user authentication for iot," pp. 1796–1802, Oct 2015.

[7] X. Yao, X. Han, X. Du, and X. Zhou, "A lightweight multicast authentication mechanism for small scale iot applications," *IEEE Sensors Journal*, vol. 13, pp. 3693–3701, Oct 2013.

[8] J. L. Hernndez-Ramos, M. P. Pawlowski, A. J. Jara, A. F. Skarmeta, and L. Ladid, "Toward a lightweight authentication and authorization framework for smart objects," *IEEE Journal on Selected Areas in Communications*, vol. 33, pp. 690–702, April 2015.

[9] H. Ren, Y. Song, S. Yang, and F. Situ, "Secure smart home: A voiceprint and internet based authentication system for remote accessing," in *2016 11th International Conference on Computer Science Education (ICCSE)*, pp. 247–251, Aug 2016.

[10] A. Ukil, S. Bandyopadhyay, A. Bhattacharyya, and A. Pal, "Lightweight security scheme for vehicle tracking system using coap," in *Proceedings of the International Workshop on Adaptive Security*, ASPI '13, (New York, NY, USA), pp. 3:1–3:8, ACM, 2013.

[11] F. Chu, R. Zhang, R. Ni, and W. Dai, "An improved identity authentication scheme for internet of things in heterogeneous networking environments," in *2013 16th International Conference on Network-Based Information Systems*, pp. 589–593, Sept 2013.

[12] L. Barreto, A. Celesti, M. Villari, M. Fazio, and A. Puliafito, "An authentication model for iot clouds," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, ASONAM '15, (New York, NY, USA), pp. 1032–1035, ACM, 2015.

[13] S. Emerson, Y. K. Choi, D. Y. Hwang, K. S. Kim, and K. H. Kim, "An oauth based authentication mechanism for iot networks," in *Information and Communication Technology Convergence (ICTC), 2015 International Conference on*, pp. 1072–1074, Oct 2015.

[14] H. Tschofenig, "Fixing user authentication for the internet of things (iot)," *Datenschutz und Datensicherheit - DuD*, vol. 40, no. 4, pp. 222–224, 2016.

[15] W. T. Su, W. M. Wong, and W. C. Chen, "A survey of performance improvement by group-based authentication in iot," in *2016 International Conference on Applied System Innovation (ICASI)*, pp. 1–4, May 2016.

[16] M. B. Barcena and C. Wueest, "Insecurity in the internet of things," vol. 1.0, March 2015.

[17] H.-P. Co., "Internet of things research study." `https://community.hpe.com/hpeb/attachments/hpeb/sws-22/1692/1/HP_IoT_Research_Study.pdf/`, July 2014.

[18] O.Bello and S.Zeadally, eds., *Next Generation Wireless Technologies: 4G and Beyond*, ch. - Communication issues in the Internet of Things, pp. 189–219. London, U.K.: Springer-Verlag, 2013.

[19] O. Bello and S. Zeadally, "Intelligent device-to-device communication in the internet of things," *IEEE Systems Journal*, vol. 10, pp. 1172–1182, Sept 2016.

[20] B. Schneier, "The Internet of Things Will Turn Large-Scale Hacks into Real World Disasters."

[21] P. F. Ovidiu Vermesan, *Building the Hyperconnected Society: Internet of Things Research and Innovation Value Chains, Ecosystems and markets*. River Publishers, 2015.

[22] I. Tashi and S. Ghernaouti-Hélie, "Security metrics to improve information security management," in *Proceedings of 6th Annual Security Conference*, 2007.

[23] S. C. Payne *A Guide to Security Metrics - SANS Institute InfoSec Reading Room*, June 2006.

[24] RedHat, "An Intelligent Systems Solution for the Internet of Things." `https://www.redhat.com/cms/managed-files/Intelligtent%20Systems%20Solution%20for%20the%20IoT.pdf/`. [Online; accessed 15-November-2016].

[25] M. V. Pawar and J. Anuradha, "Network security and types of attacks in network," *Procedia Computer Science*, vol. 48, pp. 503 – 506, 2015.

[26] A. Patel, Q. Qassim, and C. Wills, "A survey of intrusion detection and prevention systems," *Information Management & Computer Security*, vol. 18, pp. 277–290, October 2010.

[27] P. E. Ohlhausen, "In search of security metrics," *Security Management*, 2014.

[28] O. Salman, S. Abdallah, I. H. Elhajj, A. Chehab, and A. Kayssi, "Identity-based authentication scheme for the internet of things," in *2016 IEEE Symposium on Computers and Communication (ISCC)*, pp. 1109–1111, June 2016.

[29] M. Saadeh, A. Sleit, M. Qatawneh, and W. Almobaideen, "Authentication techniques for the internet of things: A survey," in *2016 Cybersecurity and Cyberforensics Conference (CCC)*, pp. 28–34, Aug 2016.

[30] D. Brumley and D. Boneh, "Remote timing attacks are practical," *Computer Networks*, vol. 48, no. 5, pp. 701 – 716, 2005. Web Security.

[31] H. La Vinh and A. R. Cavalli, "Security attacks and solutions in vehicular ad hoc networks: a survey," *International journal on AdHoc networking systems (IJANS)*, vol. 4, no. 2, pp. 1–20, 2014.

[32] S. Shunmuganathan, R. D. Saravanan, and Y. Palanichamy, "Secure and efficient smart-card-based remote user authentication scheme for multiserver environment," *Canadian Journal of Electrical and Computer Engineering*, vol. 38, pp. 20–30, winter 2015.

[33] F. Towhidi, A. A. Manaf, S. M. Daud, and A. H. Lashkari, "The knowledge based authentication attacks," in *World Congress in Computer Science*, 2011.

[34] J. Alves-Foss, "Multi-protocol attacks and the public key infrastructure," in *Proc. 21st National Information Systems Security Conference*, pp. 566–576, 1998.