Study and Analysis of different Authentication Schemes in 6LoWPAN Network

A Major Project Report Submitted in Partial Fulfillment of Requirements for the Degree of

Bachelor of Technology in Information Technology

by
Sonu Saha (BT/IT/1654)
Daisy Baruah (BT/IT/1608)
Amit Deka (BT/IT/1604)

Under the Supervision of
Dr. Subhas Chandra Sahana

Department of Information Technology
School of Technology
North-Eastern Hill University, Shillong

June 2020

Abstract

IPv6 over Low Power Wireless Personal Area Networks (6LoWPAN) is a widely used protocol in IoT communication. Sensor nodes in 6LoWPAN which helps to collect vital information from the environment and transmit to a central server through the public Internet operate with very limited resources such as limited energy, low computational power, and low memory, etc. Security functions that work in traditional internet are not suitable for these resource-constrained devices. Therefore, for the communication parties to securely communicate over 6LoWPAN, secure authentication, and session key exchange schemes which have low communication and computational overheads are designed.

The major project aims at discussing three such different authentication schemes that are suitable in the 6LoWPAN network. The three schemes viz. SAKES, EAKES6Lo, and S6AE have been considered because they provide authentication and key establishment procedure which are effective for the devices in the 6LoWPAN network.The performance analysis and comparison of all the three schemes are done based on five factors which are: storage overhead, computational overhead, communication overhead, type of cryptographic system used for the key exchange process, and the time required in the handover phase. From the analysis, it is found that S6AE is more effective than SAKES and EAKES6Lo in terms of computational, storage, and communication overhead.

The schemes are implemented in the AVISPA (Automated Validation of Internet Security Protocols and Applications) tool for formal security verification and are found to be safe. Furthermore, an informal security comparison of the schemes is done against different attacks such as replay attack, MITM, impersonation attack, IP spoofing, identity preservation attack, and jamming attack.

Keywords: 6LoWPAN, AVISPA, security, authentication and key exchange

Acknowledgements

We are grateful to God for the good health and well-being that were necessary to complete this project.

We are most grateful to our Supervisor, Dr. S.C. Sahana, Assistant Professor, Department of Information Technology. We are extremely thankful and indebted to him for sharing expertise, valuable guidance and encouragement extended to us.

We would wish to express our sincere thanks to, Prof. Goutam Saha, Head of Department, Department of Information Technology, for providing us with all the necessary facilities for the major project.

We place on record, our sincere thank you to the Dean of the School of Technology, Prof. L.J. Singh, for the continuous encouragement. We take this opportunity to express gratitude to all of the Department faculty members for their help and support. We would also like to thank our parents for the uneasing encouragement, support and attention. We are also grateful to each other for the support through this venture.

Name of the Members
Sonu Saha (BT/IT/1654)
Daisy Baruah (BT/IT/1608)
Amit Deka (BT/IT/1604)

Declaration

This is to certify that we have properly cited any material taken from other sources and have obtained permission for any copyrighted material included in this report. We take full responsibility for any code submitted as part of this project and the contents of this report.

Sonu Saha (BT/IT/1654)

Daisy Baruah (BT/IT/1608)

Amit Deka (BT/IT/1604)

Certificate

This is to certify that **Sonu Saha** (BT/IT/1654), **Daisy Baruah** (BT/IT/16-08) and **Amit Deka**(BT/IT/1604) worked in the project **Study and analysis of different Authentication Schemes in 6LoWPAN network** from February to June 2020 and has successfully completed the major project, in order to partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology under my supervision and guidance.

Dr. S.C. Sahana
Assistant Professor
Department of Information Technology
North-Eastern Hill University
Shillong-793022, Meghalaya, India

Certificate

This is to certify that **Sonu Saha** (BT/IT/1654), **Daisy Baruah**(BT/IT/16-08) and **Amit Deka** (BT/IT/1604) worked in the project **Study and analysis of different Authentication Schemes in 6LoWPAN network** from February to June 2020 and has successfully completed the major project, in order to partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology.

External Examiner

Prof. Goutam Saha

Head of Department
Department of Information Technology
North-Eastern Hill University
Shillong-793022, Meghalaya, India

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to 6LoWPAN

Low Power Wireless Personal Area Networks (LoWPANs) are an essential part of the Internet of Things (IoT) and are composed of resource-constrained devices tractable with the IEEE 802.15.4 standard. LoWPAN is a promising technology which has the potential to make many applications possible, such as e-health, smart grids, industrial automation, and environmental monitoring, to create a significant market with lots of opportunities and to bring much more benefits to humans[1]. Such networks are constricted in storage capacity, transmission range, computational capabilities, power resources, and data rate. To provide Internet connectivity to LoWPAN devices, IPv6 is considered to be the most accordant solution[2, 3]. However, IPv6 is a resource-intensive protocol originally designed for desktop and server environments and has a maximum frame size of 1280 bytes, whereas the maximum physical layer frame size for IEEE 802.15.4 is 127 bytes[4].

To make IPv6 frame size tractable with the IEEE 802.15.4 physical layer, the Internet engineering task force has standardized an IPv6 over LoWPAN (6LoWPAN) adaptation layer[5].

As shown in Figure 1.1, the 6LoWPAN protocol stack added the LoWPAN adaptation layer between the Network and IEEE 802.15.4 MAC layer. This layer provides IPv6 packet fragmentation, encapsulation, reassembly, and header compression mechanisms[4, 6]. Thus, it helps the 6LoWPAN device's direct communication with the internet by enabling transmission of IPv6 datagram over IEEE 802.15.4 links with dramatic reduction in IP overhead.

Sensor nodes deployed in a 6LoWPAN network are used to accumulate vital in-

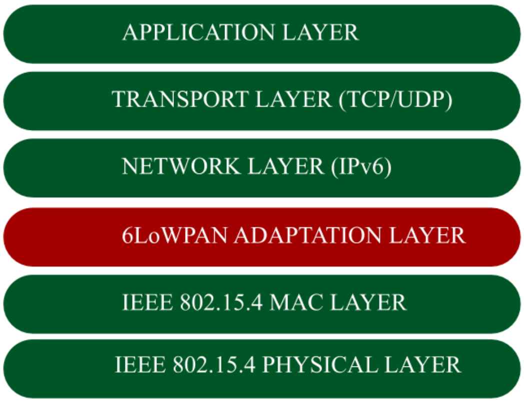


Figure 1.1: 6LoWPAN Protocol Stack

formation from surrounding environments and transmit the collected information to a central location. Thus, for an effective operation of 6LoWPANs, confidentiality and integrity of the transmitted information must be ensured. However, the original 6LoWPAN design does not include security and privacy features. In line with this, the security challenges of these devices while making the communication should be dealt in such network.

1.1.1 Security problems in 6LoWPAN

Though 6LoWPAN is a promising technology, it has various security issues unspecified, while many threats and trust crises are existing along with its development. Sensor nodes are usually deployed in an unattended environment, and messages can be easily eavesdropped in the transmission. As the 6LoWPAN devices are resource constricted, they are prone to different types of attack such as Denial-of-Service (DoS), node compromised attacks, replay attacks, etc. and traditional security schemes such as IPSec (Internet Protocol Security), IEEE 802.15.4 or ZigBee security cannot be applied to such networks because of the following reasons.

If IPSec is adopted to 6LoWPAN, it may incur more overhead to resources constrained 6LoWPAN devices. Internet Key Exchange (IKE) messaging will not work in low power networks as it incurs more signalling to the network. IEEE 802.15.4 can be used to tackle 6LoWPAN security threat but it does not clearly mentions about the key management methods to be used for the secure communication ZigBee is built on top of the IEEE 802.15.4 and could be used as one potential application level security solution to 6LoWPAN network but it has not proven its effectiveness in constrained environments despite not following the IP standard.

Therefore, in order to make IP-connectivity of the 6LoWPAN devices in reality, specialized security schemes are designed that can handle the constraints of these devices. Cryptographic encryption techniques and message validation mechanisms are applied for securing communications in 6LoWPANs. For this purpose, an Authentication and Key Exchange(AKE) scheme is essential before applying the cryptographic algorithms. An AKE scheme ensures the legitimacy of sensor nodes deployed in 6LoWPANs and also establishes a secret session key to protect communication between sensor nodes and the server from an attacker[7].

1.1.2 Application areas of 6LoWPAN

With many low power wireless sensor networks and other forms of ad hoc wireless networks, it is necessary that any new wireless system or technology has a defined area which it addresses. While there are many forms of wireless networks including wireless sensor networks, 6LoWPAN addresses an area that is currently not addressed by any other system, i.e. that of using IP, and in particular IPv6 to carry the data[8].

The overall system is aimed at providing wireless internet connectivity at low data rates and with a low duty cycle. However there are many applications where 6LoWPAN is being used:

General Automation: There are enormous opportunities for 6LoWPAN to be used in many different areas of automation.

Home automation: There is a large market for home automation. By connecting using IPv6, it is possible to gain distinct advantages over other IoT systems. The Thread initiative has been set up to standardize on a protocol running over 6LoWPAN to enable home automation.

Smart Grid: Smart grids enable smart meters and other devices to build a micro mesh network and they are able to send the data back to the grid operator's monitoring and billing system using the IPv6 backbone.

Industrial monitoring: Automated factories and industrial plants provide a great opportunity for 6LoWPAN and using automation, can enable major savings to be made. The ability of 6LoWPAN to connect to the cloud opens up many different areas for data monitoring and analysis.

1.2 Project Objective

The objective of our project is to study and implement three schemes (viz. SAKES, EAKES6Lo and S6AE) which are suitable for authentication and key establishment in 6LoWPAN network. The schemes are implemented in the AVISPA (Automated Validation of Internet Security Protocols and Applications) tool for formal security analysis. A comparative analysis is done based on factors like computational, communication, and storage overheads occurring in the authentication and key establishment process. It also presents the vulnerabilities of the schemes against different types of attacks.

1.3 Organization of the Report

The remainder of the report is organized as follows. In Chapter 2, we review authentication schemes used in 6LoWPAN systems. System models and preliminaries are discussed in Chapter 3. Chapter 4 describes the algorithm of the schemes(i.e. SAKES, EAKES6Lo and S6AE) in detail. Results and analysis of the schemes is presented in Chapter 5. Finally, the report is concluded in Chapter 6.

Chapter 2

System Models and Preliminaries

2.1 Network Model and Security Assumptions

In this project, the 6LoWPAN architecture shown in Figure 2.1 is considered. The 6LoWPAN network model consists of sensor nodes (SNs), domain router (6LDR), the access router (6LAR), and the central server (CS). SNs are used to accumulate information from the surrounding environment and transfer the collected data to CS for further processing. Moreover, 6LDR serves as a gateway and provides Internet connectivity by SNs in a domain. 6LAR provides inter-connectivity with CS on the internet. It is assumed that the communications among 6LAR, 6LDR, and CS are secure. Besides, it is assumed that CS is reachable by SNs, 6LDR, and 6LAR.

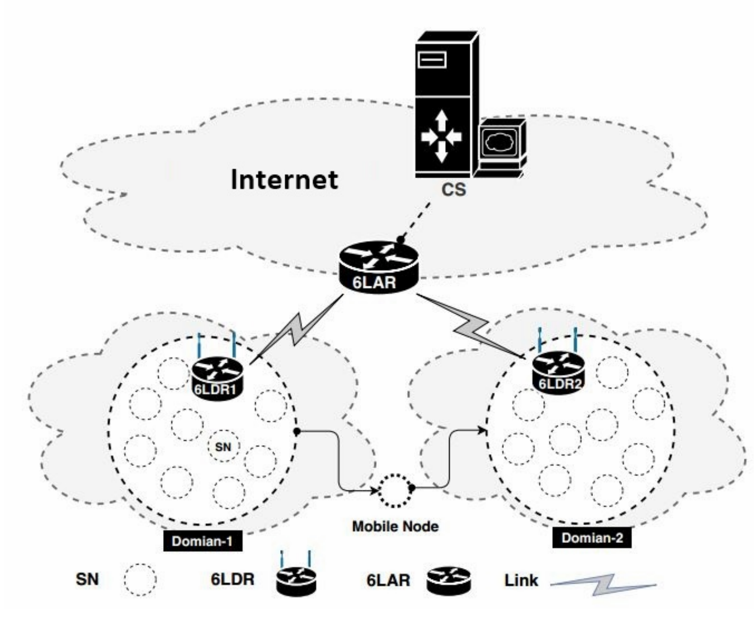


Figure 2.1: 6LoWPAN Network Architecture

2.2 Threat Model

The resource-constrained 6LoWPAN devices are usually placed in unattended and accessible locations without specific supervision and more easily attacked by adversaries compared to traditional equipment which has enough resources. The Dolev-Yao (DY) model[9] is the threat model used in the all the three schemes. The attacker may be an outsider or legitimate principal in the system that can do the following:

1. eavesdrop on any message transmitted through the network;
2. alter, decompose, synthesize, forge, replay or inject any message and sent it to a legitimate entity;
3. impersonate a legal principal;
4. decrypt or encrypt messages if obtaining the corresponding secret key;

However, it is assumed that there are certain things that an adversary is unable to do:

1. guess a nonce which is chosen from a sufficiently large space;
2. retrieve the information from a given cipher text or generate a valid cipher text from a given plain text without a complete and correct key;
3. calculate a private key that matches a given public key.

2.3 Preliminaries

2.3.1 Hash Function

Hash functions take some data of an arbitrary length (and possibly a key or password) and generate a fixed-length hash based on this input.

A cryptographic hash function must have the following properties:

- *Preimage Resistance* - Given a hash h, it should be difficult to find any message m such that h = hash(m).

- *Second Preimage Resistance* - Given an input x, it should be difficult to find another input y, where x $\neq$ y, such that hash(x) = hash(y).

- *Collision Resistance* - It should be difficult to find two different messages x and y such that hash(x) = hash(y). Such a pair is called a cryptographic hash collision. This property is sometimes referred to as strong collision resistance. It requires a hash value at least twice as long as that required for Preimage Resistance, otherwise collisions may be found by a method known as a birthday attack.

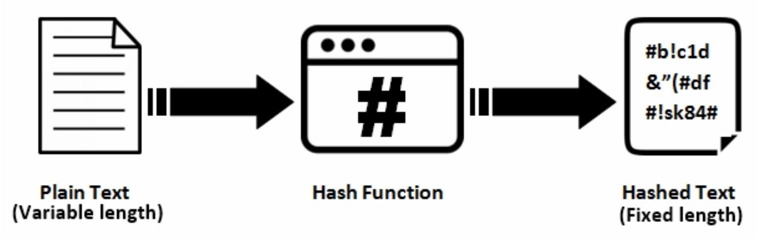


Figure 2.2: Working of Hash Function

2.3.2 Diffie Hellman Key Exchange Algorithm

Diffie Hellman (DH) key exchange algorithm is a method for securely exchanging cryptographic keys over a public communications channel. Keys are not actually exchanged – they are jointly derived. The working of the algorithm is explained below and the pictorial explanation is shown in Figure 2.3.

If Alice and Bob wish to communicate with each other, they first agree between them a large prime number p, and a generator (or base) g (where $0 < g < p$).

Alice chooses a secret integer a (her private key) and then calculates $g^{a}modp$ (which is her public key). Bob chooses his private key b, and calculates his public key in the same way.

Alice and Bob then send each other their public keys. Alice now knows a and Bob's public key $g^{b}modp$. She is not able to calculate the value b from Bob's public key as this is a hard mathematical problem (known as the discrete logarithm problem). She can however calculate $(g^{b})^{a}modp = g^{a}bmodp$.

Bob knows b and g^{a}, so he can calculate $(g^{a})^{b}modp = g^{a}bmodp$. Therefore both Alice and Bob know a shared secret $g^{a}bmodp$.

If an eavesdropper Eve was listening in on the communication knows p, g, Alice's public key $(g^a mod p)$ and Bob's public key $(g^b mod p)$. She is unable to calculate the shared secret from these values.

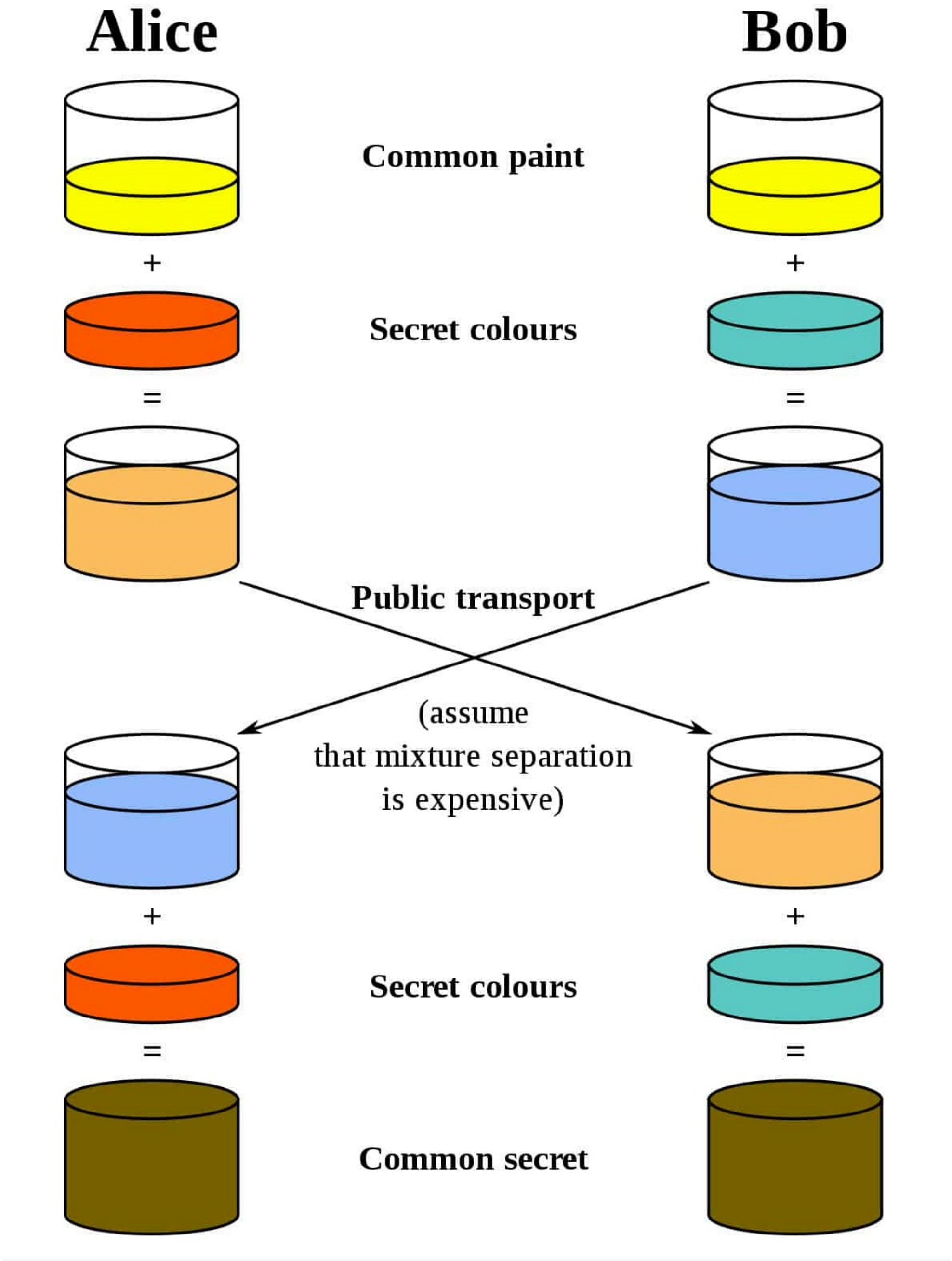


Figure 2.3: Diffie Hellman Key Exchange Algorithm

2.3.3 Elliptic Curve Cryptography

Elliptic-curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC allows smaller keys compared to non-EC cryptography to provide equivalent security[10].

An elliptic curve is the set of points that satisfy a specific mathematical equation. The equation for an elliptic curve is as below:

$$y^2 = x^3 + ax + b$$

For example, let a=3 and b=5, then when we plot the curve, it looks like this:

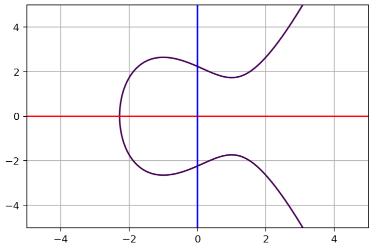


Figure 2.4: Elliptic Curve

- *Addition in ECC:*

  Lets pick two different random points with different x value on the curve, connect these two points with a straight line, let's say A and B. Then we will notice the line touches the curve at a third point. Once we find that third point and flip its y value to the other side of x axis, let's call it A+B.

  The point A+B is the sum of A and B.

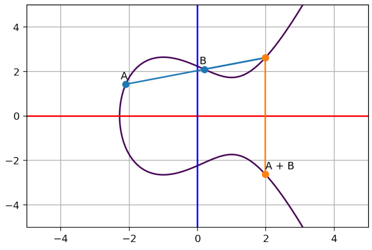


Figure 2.5: Addition of two points in Elliptic Curve

Let's pick a new point C and add to A+B to find A+B+C. We can follow the similar steps as above.

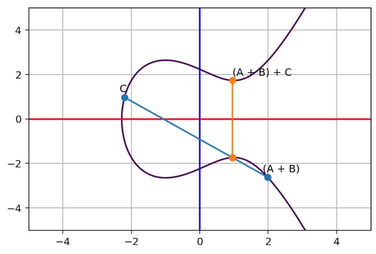


Figure 2.6: Addition of three points in Elliptic Curve

We can keep repeating the same process by adding a new point to bounce to a new sum, as long as the new line is not vertical. Adding two vertical points is an undefinable procedure. This results to what referred as the *elliptic identity* commonly denoted as ∞ (infinity). Because when we try to add two vertical points, there will never be a 3rd intersection and so we cannot define the addition in that case.

- *Point doubling in ECC:*

Take the tangent line of the current position P, and it will look like this:

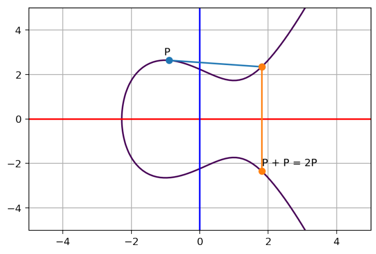


Figure 2.7: Doubling point in Elliptic Curve

We find intersection point, flip its y value to the other side of x axis, the obtained point is P+P or 2P. We can keep doing this to find coordinate for nP times.

The curve and original point P is a shared value everybody knows and agreed to use, the final point NP is our public key, safe to share to anyone. How many steps we jump, the value n is our private key. Given the curve and coordinates P and nP, it is not possible to compute n.

2.3.4 Elliptic Curve Diffie–Hellman Algorithm

Elliptic Curve Diffie–Hellman (ECDH) is a variant of the Diffie-Hellman algorithm for elliptic curves. It is actually a key-agreement protocol, more than an encryption algorithm. This basically means that ECDH defines how keys should be generated and exchanged between parties.

The working of the algorithm is as follows:

1. Both the parties agrees upon the following domain parameters:

   - The prime p that specifies the size of the finite field.
   - The coefficients a and b of the elliptic curve equation.
   - The base point G that generates our subgroup.
   - The order n of the subgroup (n is the smallest prime number such that n times G i.e. nG will lead to elliptic identity which is ∞).
   - The co-factor h of the subgroup.

2. Now, Alice and Bob generate their own private and public keys. We have the private key d_A and the public key $H_A = d_A G$ for Alice, and the keys d_B and $H_B = d_B G$ for Bob. Note that both Alice and Bob are using the same domain parameters: the same base point on the same elliptic curve on the same finite field.

3. Alice and Bob exchange their public keys H_A and H_B over an insecure channel. The Man In the Middle would intercept H_A and H_B, but won't be able to find out neither d_A nor d_B without solving the discrete logarithm problem.

4. Alice calculates $S = d_A H_B$ (using her own private key and Bob's public key), and Bob calculates $S = d_B H_A$ (using his own private key and Alice's public key). Note that S is the same for both Alice and Bob, in fact:

   $$S = d_A H_B = d_A(d_B G) = d_B(d_A G) = d_B H_A$$

   The Man In the Middle, however, only knows H_A and H_B (together with the other domain parameters) and would not be able to find out the shared secret S.

2.3.5 ASCON

ASCON is a family of authenticated encryption and hashing algorithms designed to be lightweight and easy to implement, even with added countermeasures against side-channel attacks.ASCON is a symmetric, inverse free, single pass, and online block cipher. Broadly speaking, there are two versions of ASCON: (i) ASCON-128 that takes 64 bits data block and generates 64 bits ciphertext along with 128 bits of authentication tag, and (ii) ASCON-128a that takes 128 bits data block and generates 128 bits of ciphertext along with 128 bits of authentication tag. The architecture of ASCON is given in Figure 2.8, which works under the following four stages:

1. *Initialization:* In this stage, ASCON computes the initial input to ASCON state by combining the Initialization Vector (IV), nonce, and key. The size of ASCON state is 320 bits.

2. *Associated Data (AD) Processing:* This stage processes AD that represents the data block to be transmitted in an un-encrypted form, while at the same time ensuring the integrity of the transmitted data block.

3. *Plaintext Processing:* In this stage, ASCON takes plaintext as an input and generates the ciphertext as output.

4. *Finalization:* In the final stage, ASCON generates the authentication tag, which ensures the integrity and authenticity of the ciphertext and AD.

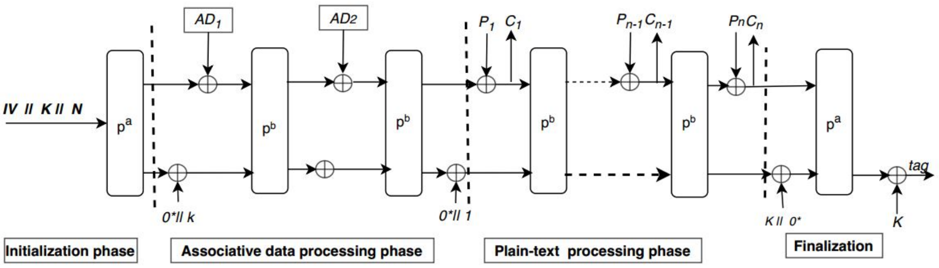


Figure 2.8: ASCON Architecture

ASCON is suitable for resource-constrained devices, such as embedded systems and radio frequency identifier tags, because of its lightweight property and minimal overheads.

2.3.6 Neighbor Discovery Protocol

Neighbor Discovery Protocol is an important protocol in IPv6. Neighbor Discovery Protocol (NDP) is based on ICMPv6 and is used to identify the relationships between different neighboring devices in an IPv6 network. Many important functions of IPv6 like resolving MAC address of an IPv6 Address (in IPv4, ARP is used for this), Router Discovery etc., are now performed using Neighbor Discovery Protocol (NDP).[11]

Internet Control Message Protocol (both ICMPv4 and ICMPv6) is a protocol which acts as a communication messenger protocol between the communicating devices in IP network. ICMP messages provide feedback, error reporting and network diagnostic functions in IP networks which are necessary for the smooth operation of IPv6.

There are five different ND messages:

- *Router Solicitation (ICMPv6 type 133):* Router Solicitation messages are requests to IPv6 Routers for Router Advertisement Messages.
- *Router Advertisement (ICMPv6 type 134):* Router Advertisements are the NDP messages generated by IPv6 Routers to advertise their presence in the link and to inform other IPv6 devices in the link about important IPv6 link parameters like network prefix, prefix length, MTU etc.
- *Neighbor Solicitation (ICMPv6 type 135):* Sent by an IPv6 device to resolve the link-layer address (MAC Address) of an IPv6 neighbor, to verify the reachability of cached link-layer address (MAC Address) and for Duplicate Address Detection (DAD)
- *Neighbor Advertisement (ICMPv6 type 136):* Neighbor Advertisement messages are response to a Neighbor Solicitation message sent from an IPv6 neighbour. An IPv6 device can also send Unsolicited Neighbor Advertisement messages to announce a change in link-layer address.
- *Redirect (ICMPv6 type 137):* Redirect messages are sent by IPv6 routers to inform IPv6 hosts in the link about a better next hop.

Neighbor Discovery Protocol (NDP) uses Router Solicitation and Router Advertisement messages (ICMPv6 Type Field Values 133 and 134 respectively) for discovering IPv6 Routers dynamically.

- *NDP (Neighbour Discovery Protocol) Router Solicitation:* IPv6 hosts multicast (to a destination All router multicast IPv6 address FF02::2) an ICMPv6 message for the key IPv6 configuration information like Default Gateway, IPv6 Prefix and Prefix Length. The ICMPv6 message which the IPv6 hosts multicasts asking for Default Router, IPv6 Prefix and Prefix Length is called as the Router Solicitation (RS) message. ICMPv6 Type value for Router Solicitation message is 133.
- *NDP (Neighbour Discovery Protocol) Router Advertisement:* IPv6 routers reply back ICMPv6 Router Advertisement (RA) message (at a destination IPv6 all nodes multicast address FF02::1) in response to a Router Solicitation message from IPv6 hosts. The Router Advertisement (RA) message contains the key IPv6 configuration information like Default Router, IPv6 Prefix, Prefix Length, link MTU etc. ICMPv6 Type value for Router Advertisement message is 134.

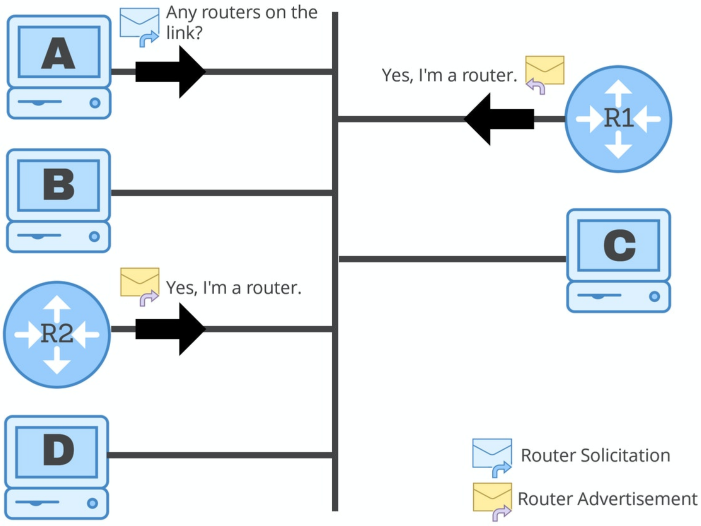
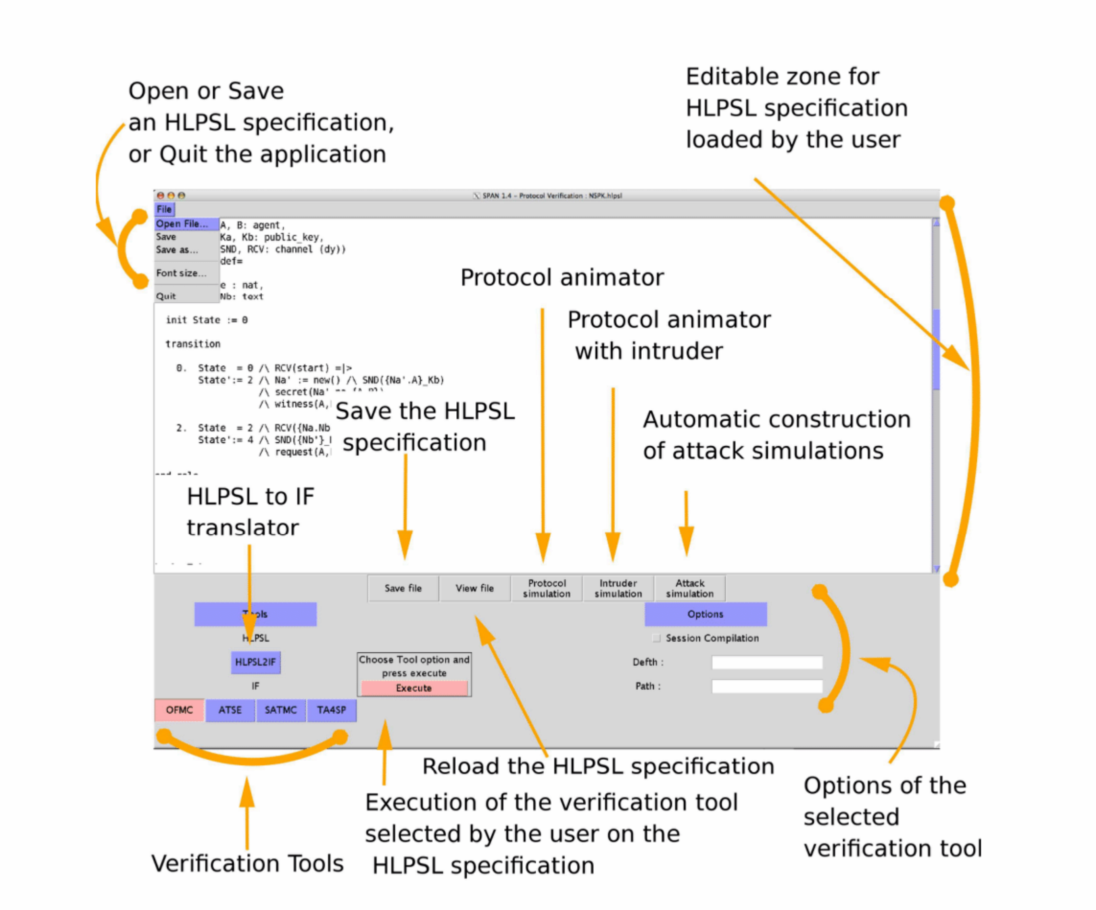


Figure 2.9: Discovering Routers : NDP

2.3.7 AVISPA

The AVISPA (Automated Validation of Internet Security Protocols and Applications) tool provides a suite of applications for building and analysing formal models of security protocols. It provides a modular and expressive formal language known as High Level Protocol Specification Language, or HLPSL for specifying protocols and their security properties, and integrates different back-ends that implement a variety of state-of-the-art automatic analysis techniques. The role of each part of the SPAN(the Security Protocol ANimator for AVISPA) tool is described in Figure 2.10



```
A, B: agent,
Ka, Kb: public_key,
SND, RCV: channel (dy))
def=

e : nat,
Nb: text

 init State := 0

 transition

   0.  State  = 0 /\ RCV(start) =|>
       State':= 2 /\ Na' := new() /\ SND({Na'.A}_Kb)
                  /\ secret(Na'
                  /\ witness(A,

   2.  State  = 2 /\ RCV({Na.Nb
       State':= 4 /\ SND({Nb'}_
                  /\ request(A,
```

Open or Save an HLPSL specification, or Quit the application
Editable zone for HLPSL specification loaded by the user
SPAN 1.4 – Protocol Verification : NSPK.hlpsl
File
Open File...
Save
Save as...
Font size...
Quit
Protocol animator
Protocol animator with intruder
Save the HLPSL specification
Automatic construction of attack simulations
HLPSL to IF translator
Save file
View file
Protocol simulation
Intruder simulation
Attack simulation
Tools
HLPSL
HLPSL2IF
IF
Choose Tool option and press execute
Execute
Options
Session Compilation
Defth :
Path :
OFMC
ATSE
SATMC
TA4SP
Reload the HLPSL specification
Execution of the verification tool selected by the user on the HLPSL specification
Options of the selected verification tool
Verification Tools

Figure 2.10: SPAN : Graphical Interface

Chapter 3

Literature Review

3.1 Related Schemes

Security is important for communication of any device with the rest of the internet. Most of the 6LoWPAN devices are battery operated and have low computation or processing power and limited storage capacity. Thus, well-known existing schemes are not suitable for implementation in networks with resource-constrained devices. To overcome the vulnerabilities of the 6LoWPAN systems, a lot of research solutions have been proposed.

To prevent the denial-of-service (DoS) attacks, which is a critical security threat to the 6LoWPANs, a novel DoS detection architecture with an intrusion detection system has been proposed in "Denial-of-service detection in 6LoWPAN based internet of things," (2013)[12] by P. Kasinathan, C. Pastrone, M. A. S. and Vinkovits, M.. The authors in "S3K: Scalable security with symmetric keys-DTLS key establishment for the Internet of Things"[13] propose a lightweight IP Security (IPsec) based scheme for 6LoWPANs to achieve secure end-to-end communication. The scheme introduces a pre-shared key concept for AKE, but it does not provide any information about the session initialization and secure mobility. The authors in "6LowPSec: An end-to-end security protocol for 6LoWPAN"[14] propose a 6LowPSec security protocol that provides end-to-end security among 6LoWPAN nodes using the existing hardware security mechanism specified by IEEE 802.15.4 Media Access Control (MAC) sub-layer. However, their proposed security protocol does not provide mobility support and header verification.

3.2 Schemes: SAKES, EAKES6Lo & S6AE

The schemes proposed in "Secure Authentication and Key Establishment Scheme (SAKES)"[15], "Enhanced authentication and key establishment scheme for M2M communications in the 6LoWPAN networks (EAKES6Lo)"[16], and "Securing 6LoWPAN using Authenticated Encryption Scheme (S6AE)"[17] have been studied in this project. These schemes provide authentication and key establishment procedure for the devices in the 6LoWPAN network. The procedures of these schemes have been discussed in detail in Chapter 4.

3.2.1 SAKES

"Secure Authentication and Key Establishment Scheme (SAKES)"[15] uses a symmetric key to encrypt messages in the authentication phase and uses a lightweight asymmetric key based on the elliptic curve cryptography (ECC) in the key establishment phase to build a session key between the 6LoWPAN devices and the server. By SAKES, the session key is calculated by using the Diffie–Hellman (DH) key exchange method, which allows two parties to have an agreement on a shared secret key by exchanging the information over an insecure channel. To reduce the computational burden on sensor nodes, SAKES performs most of the computation at gateway nodes and sends the computed key to sensor nodes in the 6LoWPAN environment. However, it has a limitation that the security protection has only provided to the static nodes in the 6LoWPAN networks.

3.2.2 EAKES6Lo

"Enhanced authentication and key establishment scheme for M2M communications in the 6LoWPAN networks (EAKES6Lo)"[16] is a hybrid cryptography approach employed for secure authentication and flexible key establishment with the consideration of the resource constraints at the 6LoWPAN nodes. A handover ticket is generated for a mobile device to achieve fast authentication when performing handovers. Besides, this proposal has an outstanding feature to provide the security supports to both of the static and mobile nodes in the 6LoWPAN networks.

3.2.3 S6AE

“Securing 6LoWPAN using Authenticated Encryption Scheme (S6AE)”[17], provides mutual authentication between the server and sensor nodes and also ensures header verification during the authentication process without employing the IPSec protocol. S6AE employs the well-known ASCON algorithm for authenticated encryption in 6LoWPANs. Additionally, S6AE employs SHA-256 hash function and bit-wise XOR operations to achieve AKE in 6LoWPANs. The proposed scheme provides end-to-end security, mobility support, and header integrity.

Chapter 4

Analysis of the undertaken Authentication Schemes

4.1 SAKES

In Secure Authentication and Key Establishment Scheme (SAKES) the authentication and key establishment (AKE) protocol has an authentication phase and a key establishment phase. It is assumed that all nodes, which have registered to the 6LBR are static. The notations used in SAKES are listed below in Table 4.1:

Table 4.1: SAKES : Notation and Description

Notation	Description
6ED	6LoWPAN End-Device
6LR	6LoWPAN Router
6LBR	6LoWPAN Border Router
S_{erv}	Server on the internet
ID	Device Identity
C_i	Cipher-text
P_{ub}	Public Key
P_{riv}	Private Key
K_{6ED6LR}	Pair wise key shared between 6ED and 6LR
$K_{6ED6LBR}$	Pair wise key shared between 6ED and 6LBR
$K_{6LR6LBR}$	Pair wise key shared between 6LR and 6LBR
N_x	Nounce
P	The prime modulus used in Session Key derivation using Diffie-Hellman
g	The generator used in Session Key derivation
MAC	Message Authentication Code

4.1.1 Secure Authentication Scheme

End device and 6LR in the 6LoWPAN must be authenticated by the 6LBR - Authentication Module (6LBR-AM) before the key establishment process is start. The 6LBR-AM checks the identity of the 6ED and 6LR to ensure secure communication of the 6LoWPAN with the remote server in the internet. For the sake of secure communication, every 6ED communicates with the nearest 6LR in a 6LoWPAN which they find using neighbor discovery protocol (NDP). Every 6ED also also sends the identity of the neighbouring 6LR as well as the remote server by using the pairwise key shared with the 6LBR. Then the 6LR forwards the message to 6LBR as every message sent by 6EDs should be individually verified by the 6LBR-AM. In the end, by examining the authenticity of the message, 6LBR-AM verifies whether the message is really sent by the 6ED and the identity of the neighbouring 6LR is correct.

The procedures to be followed in the authentication phase are explained with the following pseudo-code:

1. 6ED broadcasts HELLO request to search for neighbour 6LRS (router discovery using NDP).

2. 6ED gets HELLO response from neighbour 6LR.

3. 6ED sends the request message to the 6LBR-AM through the neighbouring 6LR.

   Create ciphertext:

   Ci = $(ID_{6ED}, ID_{6LR}, ID_{Server}, N_{6ED})K_{6ED6LBR}$

   Create MAC:

   MAC = hash(C_i,K_{6ED6LR},6ID_{6ED},N_{6ED})

   Send (MAC, Ci, ID_{6ED},N_{6ED}) to 6LR.

4. 6LR forwards the request of 6ED to the 6LBR-AM.

   Check MAC of the received message

   IF success

THEN Create MAC:

MAC = hash(C_i, $K_{6LR6LBR}$, ID_{6LR}, N_{6LR})

Send (MAC, C_i, ID_{6LR}, N_{6LR}) to 6LBR-AM.

5. 6LBR-AM checks the authenticity of the 6ED and the 6LR.

   (a) IF 6LR is registered in the 6LBR-AM

   THEN Send to 6LR:

   N_{6LBR}, $[ID_{6ED}, ID_{Server}]K_{priv_6LBR}$, $[K_{pub}, K_{priv}]K_{6LR6LBR}$

   (b) ELSE IF

   6LR is not registered in the 6LBR-AM

   THEN

   Broadcast the identity of the suspected 6LR to other nodes in the 6LoWPAN

   Look for authentic 6LR nearest to the 6ED from the registry of the 6LBR-AM

   Decide the correct path to reach the request

   Go to a

4.1.2 Key Establishment Scheme

After the 6LBR-AM identifies the 6LR and 6ED as authentic, it sends all necessary security credentials that helps the 6LR in establishing the Session Key (SK) on behalf of the 6ED with the remote server on the internet. The 6LBR-AM sends lightweight Public-Private key combination based on ECC to the 6LR which it then uses for SK establishment and communication with the server. It also sends the identity of 6LR to the 6ED using the pairwise key shared between 6LBR and 6ED.

When the remote server realizes the authenticity of the 6LR from the received message (i.e. after decryption of the cipher text encrypted with the private key of 6LBR), then the secure SK establishment process can be conducted by the server. The remote server establishes the SK by applying a one way Differ-Hellman key agreement method. The server uses its Private Key, the Public Key of 6LR, generator g and the prime number p to calculate the Session Key (SK). The SK calculation by the server (SK_{Server}) is done as follows:

$SK_{Server} = g^{K_{pub6LR}*K_{privServer}} modp$

Then the 6LR also uses its Private Key, the Public Key of the server, p and g to create the SK. The SK calculation by the 6LR is done as follows:

$SK_{6LR} = g^{K_{pubServer}*K_{priv6LR}} modp$

Finally both the parties derive $SK = SK_{Server} = SK_{6LR}$ in a secure manner. Thus, after the establishment of SK, the 6ED can make secure communication with the remote server.

The procedures for secure session key establishment between the communicating parties are described with the following pseudo-code:

1. 6LR sends request message of the following form to the server.

   Cipher-text received from 6LBR:

   Ci = $(ID_{6ED}, ID_{6LR}, ID_{Server})K_{Priv6LBR}$

   Create MAC:

   MAC = hash(C_i, ID_{6LR}, N_{6LR}, K_{Pub6LR})

   6LR sends (MAC, C_i, $6LR_{ID}$, N_{6LR})$K_{Priv6LR}$ to the server.

2. Server checks MAC of the received message

   IF success

   THEN Calculate SK based on the received message, prime modulus p and generator g.

   Create MAC:

   MAC = hash(ID_{Server}, $K_{PubServ}$, p, g, N_{Serv})

   Server sends reply message of the form:

   (MAC, ID_{Server}, p, g, N_{Serv})$K_{PrivServ}$ to 6LR.

3. On the receipt of the reply message from the server, the 6LR checks the MAC and if successful, it calculates the SK on behalf of the 6ED as the 6ED is resource constrained.

4. 6LR sends SK to the 6ED.

5. 6ED can communicate with a remote server in a secure manner by using the SK.

4.2 EAKES6Lo

The EAKES6Lo (Enhanced authentication and key establishment scheme for M2M communications in the 6LoWPAN networks) scheme is designed in the 6LoWPAN networks for the secure M2M communications.

It consists of three phases:

- predeployment phase,
- AKE phase, and
- handover phase.

The 6LoWPAN hosts (6LHs), whether it is static or mobile, need to perform the first two steps to ensure their legitimacy and prevent unauthorized disclosure of information. The mobile nodes need to follow the third step when performing handovers. It is assumed that all of the devices have a unique identity within a single LoWPAN and the link between the 6LoWPAN Edge Router(6LER) and the remote server is secure. The notation used and its description is listed in the Table: 4.2.

4.2.1 Predeployment Phase:

Before the deployment of a new 6LoWPAN node, a parameter list (*p, q, a, b, G, n, h*) of ECC is published by the system through the following steps:

1. Choose a k-bit prime number p specifying the finite field F_p.

2. Choose two coefficients a and b specifying an elliptic curve E/F_p, which is defined by the equation $y^2 = x^3 + ax + b\, modp$.

3. Choose a base point G = (x,y) on E(F_p) that generates the subgroup whose prime order is n and co-factor is h.

Table 4.2: EAKES6Lo : Notation and Description

Notation	Description
6LH	6LoWPAN Host
6LR	6LoWPAN Router
6LER	6LoWPAN Edge Router
ID_x	Identity of x
N_x	Nounce generated by x
T_x	Timestamp generated by x
T_{EXP}	Expiration timestamp generated by server
MAC	Message Authentication Code
K_{xy}	Symmetric key shared between x and y
Kpri_x	Private key of x
Kpub_x	Public key of x
HK	Secret key used for generating the MAC
SER	Server on the internet
SK_{xy}	Secret key established between x and y

After publishing the parameter list (p, q, a, b, G, n, h), a public/private key pair can be generated based on these parameters.

By the EAKES6Lo, prior to the deployment of a new node i, each node in the 6LoWPAN network has a unique identity as the mac address and must be registered with the server. The node i sends a registration request to the server with its ID and its public key. After checking that the ID has not been registered before, the server will send its public key and a randomly generated number S_i in response to the request and store ID_i, $Kpub_i$ and S_i in the database, which will be further used to produce the symmetric key between the node i and the remote server.

The symmetric key K_{Ni_SER} between the 6LoWPAN node i and the remote server is calculated by both parties in advance in order to save more time when performing the AKE phase. The key K_{Ni_SER} is computed by itself using ECDH and simple hash function through the function

$K_{Ni\_SER} = \text{Hash } (K_{pri\_Ni} * K_{pub\_SER}, S_i)$,

while key K_{SER_Ni} is obtained by the server calculating

$K_{SER\_Ni} = \text{Hash } (K_{pri\_SER} * K_{pub\_Ni}, S_i) = K_{Ni\_SER}$.

4.2.2 AKE Phase:

In this phase, the 6LHs, the 6LRs, the 6LER, and the server will exchange six messages to achieve a mutual authentication. The details of the AKE process are described as follows.

1. When a new 6LH wants to join in a LoWPAN, it will perform neighbor discovery to obtain its unique global IPv6 address. A nonce N_6LH is randomly chosen by the 6LH to be encrypted by the symmetric key K_{6LH_SER}. Then, ID_{6LH} , ID_{6LR} received through router advertisement, timestamp T_{6LH} generated by the 6LH, encrypted nonce, and the signature will be sent out to the 6LR as message 1 by the 6LH. The content of message 1 is as follows:

   $\{ID_{6LH}$,ID_{6LR},T_{6LH}, $\{N_6LH)K_{6LH\_SER}\}$ K_{pri6LH}.$\}$

2. The 6LR receives message 1 from the 6LH, will verify the signature. If the signature is valid and the identity of the router contained in the message is correct, message 1 is going to be forwarded to the 6LER as message 2.

3. When the 6LER receives the message from the 6LR, it will verify the signature of this message. If it is valid, the 6LER will compare the identities ID6LR and ID6LH with the registered ID list and check whether the identities and the addresses of the devices are consistent.

   (a) If the 6LR has not been registered or the message forwarded by the 6LR and the information related to the router contained in the message are not the same, the 6LER will inform other 6LRs and 6LHs, and add the suspected 6LR to the black list.

   When the 6LH gets the response message from the 6LER, it will find another 6LR to transmit the information.

   (b) If the 6LR is a legitimate device, the message forwarded by the 6LR, a timestamp T_{6LER}, and the value MAC1 will be signed by the 6LER and sent out to the server as message 3 to confirm that the host has access to the 6LoWPAN network. The content of message 3 is as follows:

   $\{ID_{6LH}$, ID_{6LR}, T_{6LH}, $\{N_{6LH}\}$ $K_{6LH\_SER}\}K_{pri6LH}$,

$\{T_{6LER}$, MAC1$\}$ $K_{pri6LER}$

MAC1 $= Hash_{HK}($

T_{6LER}, ID_{6LH}, ID_{6LR}, ID_{6LER}, K_{pub6LH}, K_{pub6LR}, $K_{pub6LER})$.

4. The server receives message 3 and it checks if the time of transmission is below the defined threshold using the two timestamps included in the message and the receiving time.

   Then, the information relevant to the 6LH will be retrieved from the list of registered devices. The signature signed by the 6LH and the 6LER will be verified using the corresponding public key stored in server's database. The MAC value of T_{6LER}, ID_{6LH} , ID_{6LR}, ID_{6LER}, K_{pub6LH} , K_{pub6LR}, and $K_{pub6LER}$ is compared with the received one to ensure the integrity of the received data.

   (a) If the 6LH has not been registered, a warning message will be sent to the 6LER and will be broadcasted to the whole 6LoWPAN network that the 6LH is an illegitimate device.

   (b) If the 6LH is confirmed to be legitimate, the nonce N_{6LH} can be derived from the encrypted message by symmetric key K_{SER_6LH}.

   A ticket, which is employed during the handover phase, is calculated by the function

   Ticket $= \{ID_{6LH}$, $IPv6_{6LH}$, K_{pub6LH} , T_{SER}, $T_{EXP}\}$ K_{priSER}.

   The ticket and a new MAC value computed by the ticket and the nonce N6LH are sent to the 6LER as message 4 as follows:

   $\{ID_{6LH}$, $IPv6_{6LH}$, K_{pub6LH}, T_{SER}, T_{EXP} $\}$ K_{priSER}, MAC2

   MAC2 $= Hash_H$ (Ticket—N_{6LH}).

5. When the 6LER receives the message sent by the server, it verifies the signature, stores the ID6LH in the list of the legitimate devices and forwards the message to the 6LR as message 5.

6. Similarly, the 6LR will perform the same procedure as the 6LER and send the message as message 6 to the 6LH.

7. After receiving the forwarded message from the 6LR, the 6LH checks whether the MAC value is valid and stores the ticket that can be further used when a handover occurs. Finally, the new 6LH has authenticated with the remote server and a secure connection between them has been set up.

4.2.3 Handover Phase:

When a mobile 6LH is going to perform a handover, the handover ticket issued during the authentication phase will be used to achieve a fast authentication with the 6LRs.

1. When a mobile node moves from the area of 6LR1 to the area of 6LR2, it will send a handover request to 6LR2, which contains the handover ticket, a timestamp, and the MAC3. (Ticket,T_6LH, MAC3) MAC3 = $Hash_{HK}$ (Ticket—T_{6LH}).

2. Upon receiving the request from the 6LH, the 6LR2 should make sure whether the ticket is expired.

   If the ticket is expired, the request is discarded. Else, the 6LR2 verifies the signature and the value of MAC3. If they are valid, a message will be sent to the server for authentication.

3. When the server receives the message, it does the following :

   (a) it will verify the authenticity of 6LH and 6LR2.

   (b) then, a notification will be sent to the 6LR1 to inform that the 6LH is no longer in its area.

   The 6LR1 checks if the 6LH is out of its range and deletes the device from its member list. An acknowledge message (ACK) will be sent back to the server.

   (c) When receiving the confirmation from the 6LR1, a random number will be generated by the server for establishing a symmetric key shared between the 6LH and the 6LR2. The content of the message sent to the 6LR2 is as follows:

   $\{ID_{6LH}$, ID_{6LR2}, T_{SER}, MAC4$\}$ K_{priSER}, $\{N_s\}$ K_{6LR2_SER},

$\{N_s\}\ K_{6LH\_SER}$

MAC4 = $Hash_{HK}\ (ID_{6LH},\ ID_{6LR2},\ T_{SER},\ N_s)$. .

4. After receiving the message, the random number will be stored by the 6LR2 if the signature, timestamp, and the MAC value are valid and the rest of the messages will be transmitted to the 6LH.

5. Similarly, the 6LH verifies the validity of the message and stores the nonce N_s. Then, the symmetric key shared between the 6LH and the 6LR2 can be calculated as follows:

   K_{6LH_6LR2}

   = Hash(K_{pri_6LH} * $K_{pub6LR2}$,N_s)

   = Hash($K_{pri6LR2}$ * K_{pub6LH} ,N_s)

   = K_{6LR2_6LH}

4.3 S6AE

Securing 6LoWPAN using Authenticated Encryption (S6AE) Scheme verifies the legitimacy of SNs at the CS, and validates the integrity and authenticity of messages exchanged between SNs and the CS in 6LoWPANs.

In S6AE, after verifying the authenticity of SNs, CS and SNs establish secret keys using ASCON as the encryption scheme. SHA-256 is used to generate unique output strings by using the S6AE secret parameters, and bit-wise XOR operations are used to reduce the computational and storage costs.

S6AE consists of the three phases:

- registration phase,
- the AKE phase, and
- the handover phase.

The 6LoWPAN sensor nodes (SN), whether it is static or mobile, need to perform the first two phases i.e. the registration phase and the AKE phase whereas only a mobile SN requires to execute the handover phase.

The notations used in S6AE are listed below in Table 4.3:

Table 4.3: S6AE : Notation and Description

Notation	Description
CS,SN	Central Server and 6LoWPAN Sensor node
6LDR	6LoWPAN Domain Router
6LAR	6LoWPAN Access Router
SID_x	Pseudo-identity of x
ID_x	Secret real identity of x
SP_1, SP_1^n	Secret parameter used in authentication process
T_x	Timestamp at x
E_k(x)	Encryption of message "x" using the secret-key "k"
D_k(x)	Decryption of message "x" using the secret-key "k"
$\langle Tag_x,\ Tag'_x\ \rangle$	Authentication parameter generated by encryption and decryption algorithm at x
$\langle IV_x,\ IV'_x\ \rangle$	Initialization vectors at at x
$\langle S_k,\ S'''_k\ \rangle$	ASCON initialization states at SN
$\langle S'_k,\ S''_k\ \rangle$	ASCON initialization states at CS
$S_k^h,\ S_k'^h$	Initialization states at CS and SN in the handover phase, respectively
K_sn	Keys for SN
$R_n,\ R_{s1},\ R_{s2}$	Random number used in authentication process
T_h	Timestamp used in handover phase
R_h	Random number used in handover phase
H	Cryptographic Hash function
$\oplus$	Bit-wise XOR operator
$\|\|$	Concatenation
MAC_x	MAC address of x

4.3.1 Sensor Registration Phase

This phase deals with the registration of SN before its deployment in 6LoWPAN. CS performs the following operations to register SNs. It

1. calculates the master key K_m by computing K_m = H(ID_{cs} $\|$ r_{cs}), where ID_{cs} is the real identity of CS and r_{cs} is a random number.

   CS divides K_m into four equal chunks of 64 bits, namely K_m^1, K_m^2, K_m^3, and K_m^4, and computes $K_{cs} = K_m^1 \oplus K_m^2 \oplus K_m^3 \oplus K_m^4$, where K_{cs} is a temporary key for CS.

2. assigns a unique ID_{sn} of 64 bits for SN.

3. picks a key K_{sn} of 64 bits for SN and computes the pseudo-identity SID_{sn} = $ID_{sn} \oplus K_{sn} \oplus K_{cs}$.

4. computes $H_r = \mathrm{H}(K_m \parallel K_{sn} \parallel ID_{sn})$ and derives security parameter SP_1 by computing $SP_1 = H_r^1 \oplus H_r^2 \oplus H_r^3 \oplus H_r^4$, where H_r^1, H_r^2, H_r^3, and H_r^4 are four equal chunks of 64 bit H_r.

Finally, CS stores SN related secret information, i.e., $\{ID_{sn}$, SP^1, K_{sn}, K_{cs}, $MAC_{sn}\}$ into its database and $\{ID_{sn}$, SP^1, SID_{sn}, K_{sn}, $MAC_{cs}\}$ in the memory of SN while making use of a secure channel. CS also stores SID_{sn} into 6LDR memory through a secure channel.

4.3.2 Sponge State Generation

The initialization phase S_k of ASCON consists of 320 bits, known as initialization states S_k. In the proposed scheme, S_k can be derived as follows. SN

1. generates a random number R1 of 64 bits and time stamp T_{sn} of 32 bits,

2. computes $IV_{sn} = R_1 \parallel SID_{sn}$, where IV_{sn} is an initialization vector for SN,

3. computes $H_s = \mathrm{H}(ID_{sn} \parallel SID_{sn} \parallel SID_{ldr} \parallel T_{sn})$ and derives $S_k = IV_{sn} \parallel H_s^{24}$, where H_s^{24} is the first 24 bytes of H_s. The size of S_k is 320 bits (H_s24 = 24 bytes + IV_{sn} = 16 bytes), which is served as input to the encryption algorithm during the initialization phase.

4.3.3 Associative Data Generation

The following operations are performed to generate AD:

1. SN computes $H_{ad} = \mathrm{H}(IF_{sn} \parallel \mathrm{G6} \parallel \mathrm{GC} \parallel MAC_{sn})$. It then divides H_{ad} into two equal parts, i.e., H_{ad}^1 and H_{ad}^2 each of 128 bits.

2. SN computes AD $= H_{ad}^1 \oplus H_{ad}^2$ and divides AD into two equal parts, i.e., AD_1 and AD_2, each of 64 bits.

3. The encryption algorithm takes AD_1 and AD_2 as the inputs at the associative data processing phase to preserve their integrity.

4.3.4 Authentication and Key Exchange

In this phase, SN achieves the anonymous authentication and key agreement with CS via the intermediate nodes, 6LDR and 6LAR. After establishing a secret key, SN and CS can exchange data securely. S6AE exchanges four messages to accomplish the authentication process. The detail of the messages exchanged in the proposed scheme is given in the following steps:

1. SN generates a random number R_{s1} of 64 bits and timestamp Tsn of 32 bits for computing X = $ID_{sn} \oplus R_{s1} \oplus SP_1$, and Y = $ID_{sn} \oplus R_{s1}$, where the sizes of X and Y are 64 bits.

   The encryption algorithm takes S_k as shared secret inputs during the initialization phase, $\langle AD_1$, AD_2 $\rangle$at the associative data processing phase, $\langle$X $\|$ Y $\rangle$at the plaintext processing phase, and produces ciphertext $C_1 = E_{Sk}\{AD_1$, AD_2, $\langle$X $\|$ Y $\rangle\}$ and Tag_{sn} that is generated automatically by ASCON.

   C_1 ensures the confidentiality of the plaintext $\langle$X $\| Y$ $\rangle$. The generated Tag_sn guarantees the authenticity and integrity of the ciphertext C_1 at the receiving end. Tag_{sn} provides the same functionality as Message Authentication Code (MAC).

   SN also computes Z = $SID_{sn} \oplus SID_{ldr}$, where SID_{ldr} is the temporary identity of 6LDR. After performing the above operations, SN constructs a message M1 : $\langle T_{sn}$ $\|$ Z $\|$ $\langle C_1$ $\|$ Tag_{sn} $\rangle\|$ R_1 $\rangle$and forwards it to 6LDR to be processed further.

2. After receiving M1 from SN, 6LDR picks out Z from the received message and computes SID_r = Z $\oplus$ SID_{sn}. 6LDR compares SID_r with the stored SID_{ldr} in its memory. If the contents of both the SID_r and SID_{ldr} are the same, 6LDR appends its SID_{ldr} with the received M1 for generating and forwarding the new message M2 : $\langle SID_{ldr}$ $\|$ M1 $\rangle$to 6LAR. Contrarily, 6LDR aborts the AKE process and sends an error message back to SN.

3. 6LAR receives the newly generated M2 from 6LDR and checks SID_{ldr} in the current list of the registered devices. If 6LAR does not find SID_{ldr} in the list, it will abort the AKE process and add unverified SID_{ldr} in the blacklist.

On the contrary, upon successful verification of the SID_{ldr} for M2, 6LAR picks a timestamp T_{lar} and computes H_{lar} = H(M2 $\|$ SIDlar $\|$ T_{lar} $\|$ K_{lar}), where K_{lar} is the pre-shared key between 6LAR and CS, and SID_{lar} is the temporary identity of 6LAR. 6LAR then generates and forwards message M3: $\langle SID_{lar}$ $\|$ T_{lar} $\|$ M2 $\|$ H_{lar} $\rangle$to CS for further processing.

4. Upon receiving M3 from 6LAR, CS retrieves secret information related to 6LAR, such as a K_{lar} using SID_{lar}.

   It checks the validity of T_{lar} by verifying if M3 is received within the maximum transmission delay (T_d) limit by computing $T_d \geq T_r$ - T_{lar}, where T_r is the received timestamp of M3. To verify the integrity of M3, CS computes $H_l'ar$ = H(M2 $\|$ SID_{lar} $\|$ T_{lar} $\|$ K_{lar}).

   If the computed $H_l'ar$ and the received H_lar are not identical, CS aborts the AKE process and adds 6LAR to the current list of fake devices. After checking the integrity of M3, CS retrieves M2 from M3, and checks if the condition $T_d \geq T_r$ - T_{sn} holds. If the condition does not hold, then CS rejects M2. Moreover, CS also checks whether a valid SID_{ldr} exists in the current list of 6LDR devices.

   On successful verification of SID_{ldr}, CS picks Z from M2, derives SID_{sn} by computing $SID_{ldr} \oplus$ Z, and checks if SID_{sn} exists in its database. After the verification of the SID_{sn}, CS retrieves the information stored in its database, such as ID_{sn}, K_{cs}, K_{sn}, and SP_1.

5. CS generates IV_{cs} by concatenating R_1 with SID_{sn}, which are attached with the received M2. CS also computes H_s' = H(ID_{sn} $\|$ SID_{sn} $\|$ SID_{ldr} $\|$ T_{sn}) to derive S_k'. Moreover, CS determines AD by using the received header information and the stored MAC_{sn} in CS's database by computing H'_{ad} = H(IV_{sn} $\|$ G6 $\|$ GC $\|$ MAC_{sn}), $AD_x = {H'}^1_{ad} \oplus {H'}^2_{ad}$ and divides AD_x into two parts, i.e., AD'_1 and AD'_2. AD is the input to the encryption algorithm and its purpose is to ensure the integrity of header information.

   In addition, CS performs the decryption operation D'_{sk} { $\langle AD'_1$, AD'_2, $C_1\}$ $\rangle$,, where S'_{sk} is the input at the initialization phase, AD'_1 and AD'_2 are the inputs at associative data processing phase, and C_1 is the input at the ciphertext processing phase. Moreover, the decryption algorithm generates Tag_g before extracting the plain-text information. ASCON generates the authentication tag automatically after processing AD and ciphertext.

Then CS checks the condition $Tag_{sn} = Tag_g$ where Tag_{sn} is received with M1. An inverse free authenticated encryption scheme generates the same authentication tag during the encryption and decryption process, if there is no modification in AD and ciphertext. However, if there is any modification in the communicated message, the generated authentication tag will be different, which causes the failure of authentication process in the proposed AKE. If the condition holds, decryption process will reveal the plain-text information. Otherwise, CS will abort the AKE process.

The revealed plain-text, after the decryption of C1, includes X and Y. CS picks the retrieved ID_{sn} and performs $ID_{sn} \oplus$ Y operation to determine R_{s1} for computing $SP'_1 = ID_{sn} \oplus R_{s1} \oplus$ X. Furthermore, in order to check the legitimacy of SN, CS checks the condition $SP_1 = SP'_1$. If the condition holds, CS registers SN as a legitimate device, otherwise, CS will abort the AKE process.

6. After verifying the legitimacy of SN, CS picks timestamps Ts of 32 bits. CS picks three random numbers R_{s2}, R_2, and R_n each of 64 bits. CS then computes $H"_r$ = H($K_{cs} \parallel R_n \parallel ID_{sn}$) and calculates a new security parameter SP^n_1 by computing $SP^n_1 = H"^1_r \oplus H"^2_r \oplus H"^3_r \oplus H"^4_r$, where $H"^1_r$, $H"^2_r$, $H"^3_r$, $H"^4_r$ are four equal chunks of $H"_r$ each of 64 bits. CS calculates Y1 = R_n $\oplus$ K_{cs}, X1 = Y1 $\oplus$ R_{s1}, and $IV'_{cs} = R_2 \parallel X_1$, where IV'_{cs} is the initialization vector at CS and R_2 is the random number of 64 bits.

   To generate $S"_k$, CS computes $H"_s$ = H($ID_{sn} \parallel R_{s1} \parallel T_s \parallel T_{exp} \parallel Y_1$), where the size of $H"_s$ is 256 bits and calculates $S"_k = H"^{24}_s \parallel IV_{cs}$, where $H"^{24}_s$ are the first 24 bytes of $H"_s$. Next, CS calculates AD by computing $H"_{ad}$ = H($IV_{cs} \parallel$ G6 $\parallel$ GC $\parallel$ MAC_{cs}), $AD_{x1} = H"^1_{ad} \oplus H"^2_{ad}$ and divides AD_{x1} into two parts, i.e., $AD"_1$ and $AD"_2$. For secure communication in future, CS computes a session key K_{se} by calculating K_{se} = H($ID_{sn} \parallel$ Y1 $\parallel SP^n_1 \parallel R_{s1}$ $\parallel {}_{s2}$). Moreover, for secure handover from one domain to another domain, CS calculates a unique ticket T_{ic}^{sn} for SN by computing $T_{ic}^{sn} = ID_{sn} \oplus R_{s2}$ $\oplus$ R_{s1} $\oplus$ Y1 $\oplus$ SP_1^n). SN will make use of the generated T_{ic}^{sn} during the handover process. CS also picks T_{ic}^{sn}'s expiry time T_{exp} (32 bits).

In addition, the encryption algorithm takes into account $S"_k$ during the initialization phase, $AD"_1$ and $AD"_2$ during the associative data processing phase, and $\langle SP_1^n \parallel R_{s2} \rangle$during the plain-text information processing phase, in order to generate $C_2 = E_{S"k}$ { $AD"_1$, $AD"_2$, $\langle SP_1^n \parallel R_{s2} \rangle$} and Tag_{cs}. Moreover, CS constructs the message M4: $\langle T_{cs} \parallel T_{exp} \parallel$ X1 $\parallel \langle C_2 \parallel Tag_{cs} \rangle \parallel R_2 \rangle$, and forwards it to 6LAR. 6LAR and 6LDR simply relay M4 to SN. Furthermore, CS stores the parameters $\{ID_{sn}, SP_1, SP_1^n, K_{cs}, T_{ic}^{sn}, T_{exp}\}$ in its memory.

7. After receiving M4, SN checks the validity of timestamp T_s by checking the condition $T_d \geq T_r$ - T_{sn}, where T_d is the maximum allowed time TD and T_r is the period in which M4 is received. Significantly, SN will reject M4 if T_s exceeds the maximum allowed delay.

   SN picks R_2, X1 from the received M4 and calculates $IV'_{sn} = R_2 \parallel$ X1. SN also computes Y1 = $R_{s1} \oplus$ X1, H'''_s = H($ID_{sn} \parallel R_{s1} \parallel T_s \parallel$ Y1) and S'''_k = ${H'''}_s^{24} \parallel {IV'}_{sn}$, where ${H'''}_s^{24}$ is the first 24 bytes of H'''_s.

   Next, SN calculates AD by computing H'''_{ad} = H($IV_{cs} \parallel$ G6 $\parallel$ GC $\parallel$ $MACcs$), $AD_{x2} = {H'''}_{ad}^{1} \oplus {H'''}_{ad}^{2}$ and divides AD_{x2} into two parts, i.e., AD_1''' and AD_2'''. The decryption algorithm takes S_k''' as the input during the initialization phase, AD_1''' and AD_2''' during the associative data processing phase, C_2 during the ciphertext processing phase, and performs the decryption operation $D_{S'''k}$ { AD_1''', AD_2''', C_2}, to generate Tag'_{sn}.

   In the final step, SN checks the condition $Tag_{cs} = Tag'_{sn}$. If the condition holds then decryption algorithm will reveal the plain-text information, i.e., $\langle SP_1^n \parallel R_{s2} \rangle$. Additionally, SN computes the session key K_{se} by computing K_{se} = H($ID_{sn} \parallel$ Y1 $\parallel SP_1^n \parallel R_{s1} \parallel R_{s2}$) to secure future communications with CS. In addition, SN calculates a unique ticket $T_{ic}^{sn} = ID_{sn} \oplus R_{s2} \oplus R_{s1}$ $\parallel Y_1 \parallel SP_1^n$), which will be used during the handover process. Finally, SN stores the parameters $\{ID_{sn}, SP_1^n, SID_{sn}, K_{sn}, T_{ic}^{sn}, T_{exp}\}$ in its memory.

4.3.5 Handover Phase

A sensor node can move from network one domain(say Domain 1) to another domain (say Domain 2). Hence, it is essential to verify the authenticity of a roaming SN with minimal overhead complexity. SN utilizes the ticket T_{ic}^{sn}, generated during the AKE phase, to accomplish fast authentication. SN performs the following operations during the handover process.

1. When an SN moves from the communication range of 6LDR1 in Domain-1 to the communication range of 6LDR2 in Domain-2, SN sends a handover request to 6LDR2.

   SN checks T_exp of T_{ic}^{sn}, which is stored in SN's memory. If T_{ic}^{sn} is not expired then SN picks the timestamp T_h and computes H_h = H(T_{ic}^{sn} $\|$ T_h $\|$ SID_{sn}).

   SN then constructs a message M_{h1}: $\langle SID_{sn} \parallel T_h \parallel T_{ic}^{sn} \parallel H_h\ \rangle$and forwards M_{h1} to 6LDR2.

   6LDR2 checks if T_h is fresh or not. To check integrity of M_{h1}, 6LDR2 computes H_{h2} = H(T_{ic}^{sn} $\|$ T_h $\|$ SID_{sn}) and checks the condition $H_{h2} = H_h$. If the condition holds, 6LDR2 stores SID_{sn} in its memory and forwards M_{h1} to CS. Contrarily, it aborts the handover process and adds SIDsn into blacklist in its database.

   After receiving M_{h1}, CS computes H_{h3} = H(T_{ic}^{sn} $\|$ T_h $\|$ SID_{sn}) and checks the condition $H_{h3} = H_h$. If the condition holds, CS checks if SID_{sn} exists in its database and verifies the condition $T_{ic}^{sn} = T_{ic}^{sn}$. If the condition holds, CS continues the handover process, otherwise CS marks ID_{sn} as a compromised node and broadcasts ID_{sn} in the network. CS also sends a message to 6LDR1 to delete SID_{sn} from its memory. 6LDR1 sends an acknowledgment to CS. T_{ic}^{sn} is the stored ticket at SN and CS.

2. CS picks two random numbers R_n, R_1 each of 64 bits, and timestamps T_{exp}^{n} and T_{h1} each of 32 bits. It also computes $S_k^k = (K_{es} \parallel R_h \oplus ID_{sn})$ and P $= R_n \oplus SP_1^n$, where the size SP_1^n is 64 bits. CS calculates $C_h = E_{S_k^k}$ (P $\|$ T_{exp}^{n} $\|$ T_{h1}) and Tag_{cs} by using the encryption algorithm. The Tag_{cs} ensures the authenticity of the transmitted information. It also computes the new

session key as K_{se}^{n} = H(ID_{sn} $\|$ R_n $\|$ K_{se}). CS constructs a message M_{h2}: $\langle SID_{sn}$ $\|$ C_h $\|$ Tag_{cs} $\rangle$and forwards M_{h2} to 6LDR2. Upon receiving M_{h2}, 6LDR2 looks up SID_{sn} in 6LDR2's memory. If SID_{sn} exists in the memory of 6LDR2, 6LDR2 forwards M_{h2} to SN.

3. After receiving the message M_{h2} from CS, SN performs the decryption using $D_{S'^{k}_{h}}$ $\{C_h$ $\}$, where S'^{k}_{h} = (K_{es} $\|$ R_h $\oplus$ ID_{sn}).

   The decryption process reveals the plain-text, which is (P $\|$ T_{exp}^{n} $\|$ T_{h1}) and also it generates the Tag_{sn}. SN checks the condition $T_d \geq T^r$ - T_h. If the condition holds then SN considers M_{h1} valid, otherwise it rejects M_{h1}. T_{exp}^{n} indicates new expiry time of the T_{ic}^{sn}.

   SN checks the condition Tag_{sn} = Tag_{cs}. If the condition holds, then SN computes R'_n = SP_1^n $\oplus$ P and SP'^{n}_{1} = P $\oplus$ R'_n.

   Authentication will be successful if the stored SP_1^n and the computed SP'^{n}_{1} are the same. SN generates a symmetric key between SN and CS by computing K_{se}^{n} = H(ID_{sn} $\oplus$ R_n $\oplus$ K_{se}).

   Finally, SN replaces the stored session key K_{se} with the new session key K_{se}^{n} in the memory and updates the expiry time T_{exp}^{n} in the tuple $\{$ ID_{sn}, SP_1^n, SID_{sn}, K_{se}^{n} , T_{ic}^{sn}, T_{exp}^{n} $\}$.

Chapter 5

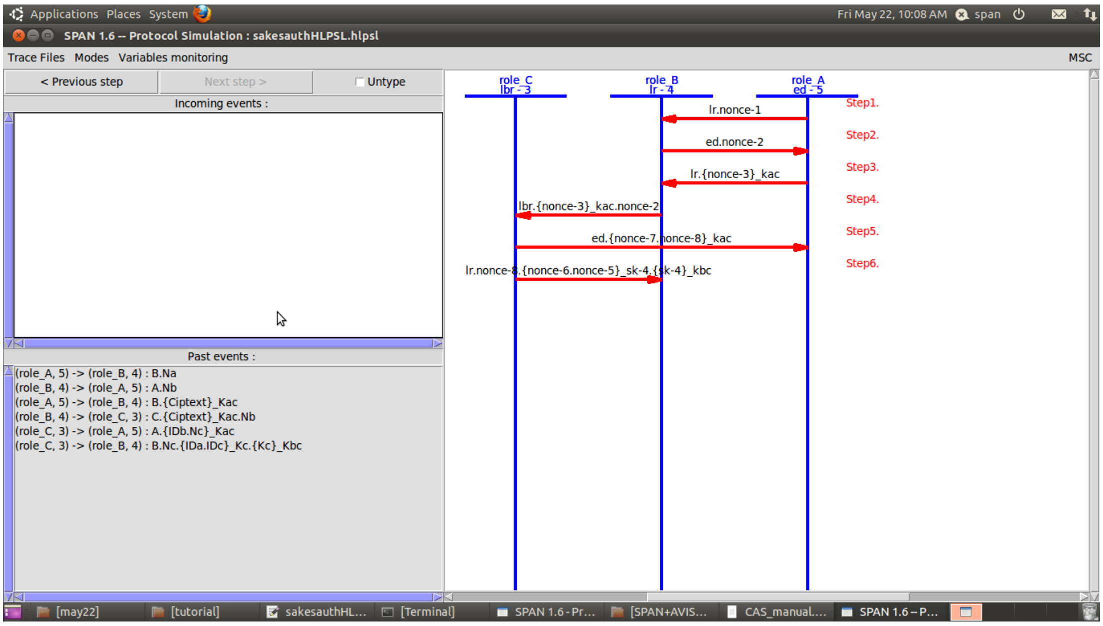
Simulation and Result Analysis

5.1 Protocol Simulation

The following simulations of the scheme is done using AVISPA tool.

5.1.1 SAKES

1. *Authentication Phase*



```
lr.nonce-1
ed.nonce-2
lr.{nonce-3}_kac
lbr.{nonce-3}_kac.nonce-2
ed.{nonce-7.nonce-8}_kac
lr.nonce-8.{nonce-6.nonce-5}_sk-4.{sk-4}_kbc
```

```
(role_A, 5) -> (role_B, 4) : B.Na
(role_B, 4) -> (role_A, 5) : A.Nb
(role_A, 5) -> (role_B, 4) : B.{Ciptext}_Kac
(role_B, 4) -> (role_C, 3) : C.{Ciptext}_Kac.Nb
(role_C, 3) -> (role_A, 5) : A.{IDb.Nc}_Kac
(role_C, 3) -> (role_B, 4) : B.Nc.{IDa.IDc}_Kc.{Kc}_Kbc
```

Figure 5.1: SAKES : Authentication Phase

Step 1: 6ED → 6LR : Hello (N_{6ED})

Step 2: 6LR → 6ED : Hello (N_{6LR})

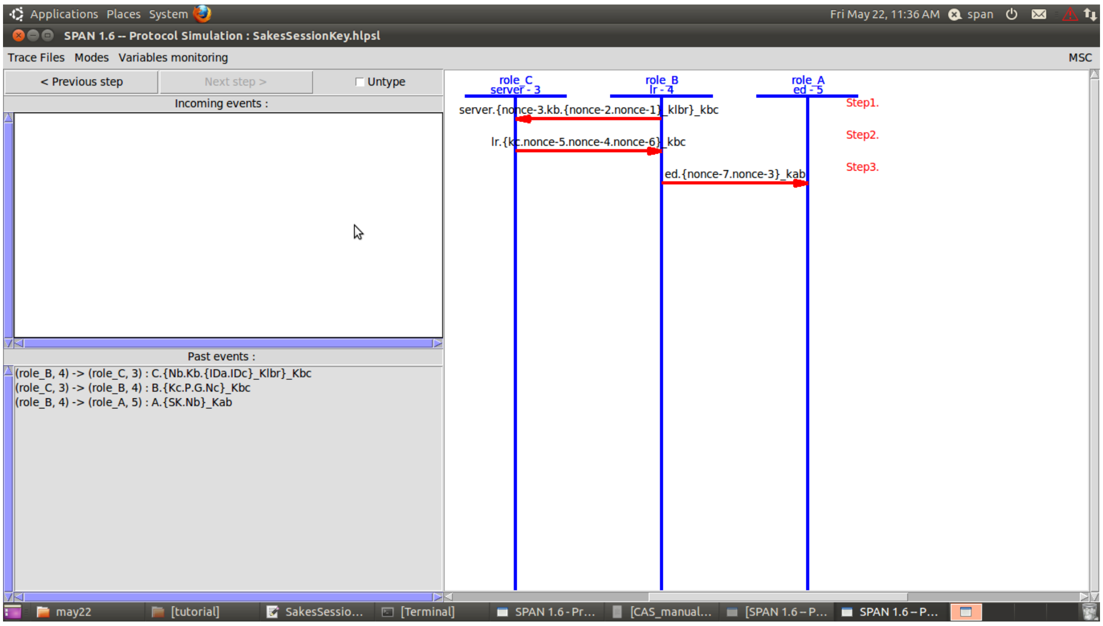
Step 3: 6ED → 6LR : $(ID_{6ED},\ ID_{6LR},\ ID_{Serv},\ N_{6ED})\ K_{6ED-6LBR}$

Step 4: 6LR → 6LBR-AM : Forward 6ED request (N_{6LR})

Step 5: 6LBR-AM → 6LR : $\{N_{6LBR},\ [ID_{6ED},\ ID_{6LR},\ ID_{Serv}]\ K_{Priv6LBR},\ (K_{pub},\ K_{pri})\}K_{6LR-6LBR}$

Step 6: 6LBR-AM → 6ED : $(ID_{6LR},\ N_{6LBR})$

2. *Key Establishment Phase*



```
(role_B, 4) -> (role_C, 3) : C.{Nb.Kb.{IDa.IDc}_Klbr}_Kbc
(role_C, 3) -> (role_B, 4) : B.{Kc.P.G.Nc}_Kbc
(role_B, 4) -> (role_A, 5) : A.{SK.Nb}_Kab
```

Figure 5.2: SAKES : Key Establishment Phase

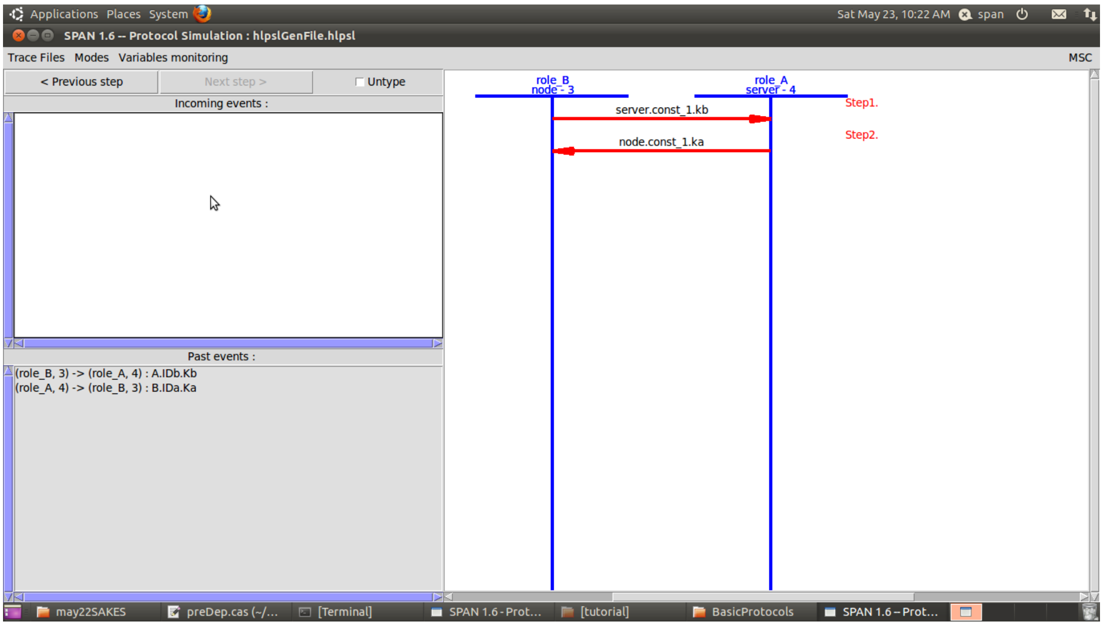
Step 1: 6LR → Server: Request $\{\ N_{6LR},\ K_{pub6LR},\ [ID_{6ED},\ ID_{Serv},]\ K_{priv6LBR}\ \}\ K_{priv6LR}$

Step 2: Server → 6LR: Reply $\{\ K_{pubServ},$ P, g, $N_{Serv},\ \}\ K_{privServ}$

Step 3: 6LR → 6ED: (SK, N_{6LR}) $K_{6ED-6LR}$

5.1.2 EAKES6Lo

1. *Registration Phase*



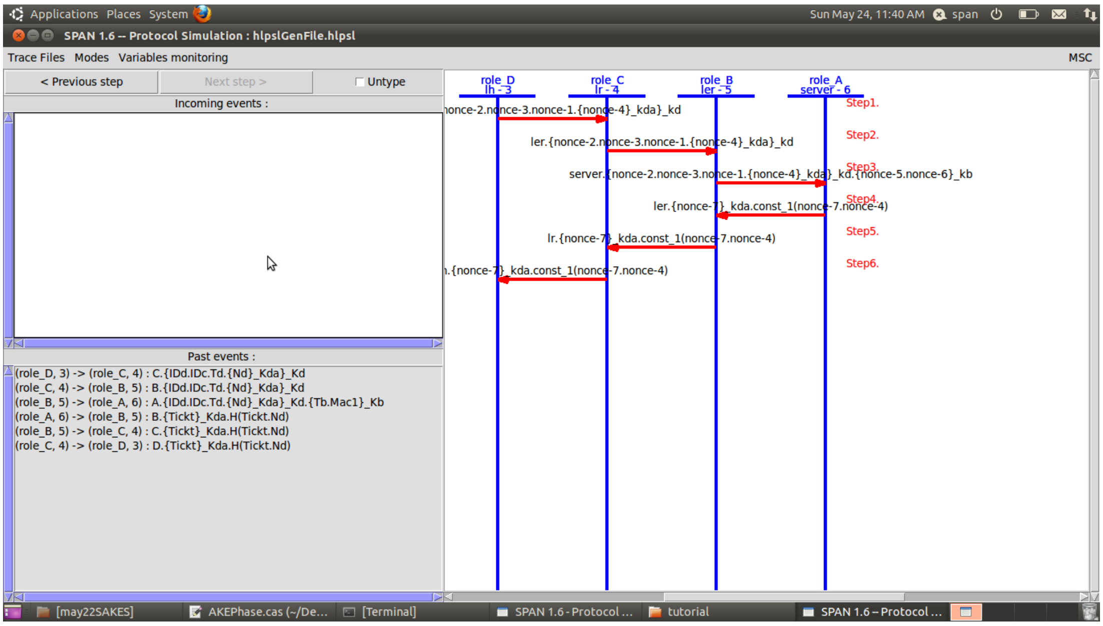
```
(role_B, 3) -> (role_A, 4) : A.IDb.Kb
(role_A, 4) -> (role_B, 3) : B.IDa.Ka
```

Figure 5.3: EAKES6Lo : Registration Phase of node

Step 1: Node i → Server: Registration Request $ID_i \parallel K_{pubi}$

Step 2: Server → Node i: Registration Response $S_i \parallel K_{pubs}$

2. *AKE Phase*


Applications Places System
Sun May 24, 11:40 AM span
SPAN 1.6 -- Protocol Simulation : hlpslGenFile.hlpsl
Trace Files Modes Variables monitoring
MSC
< Previous step
Next step >
□ Untype
Incoming events :
Past events :
role_D lh - 3
role_C lr - 4
role_B ler - 5
role_A server - 6
Step1.
Step2.
Step3.
Step4.
Step5.
Step6.
nonce-2.nonce-3.nonce-1.{nonce-4}_kda}_kd
ler.{nonce-2.nonce-3.nonce-1.{nonce-4}_kda}_kd
server.{nonce-2.nonce-3.nonce-1.{nonce-4}_kda}_kd.{nonce-5.nonce-6}_kb
ler.{nonce-7}_kda.const_1(nonce-7.nonce-4)
lr.{nonce-7}_kda.const_1(nonce-7.nonce-4)
.{nonce-7}_kda.const_1(nonce-7.nonce-4)
[may22SAKES]
AKEPhase.cas (~/De...
[Terminal]
SPAN 1.6 - Protocol ...
tutorial
SPAN 1.6 - Protocol ...

```
(role_D, 3) -> (role_C, 4) : C.{IDd.IDc.Td.{Nd}_Kda}_Kd
(role_C, 4) -> (role_B, 5) : B.{IDd.IDc.Td.{Nd}_Kda}_Kd
(role_B, 5) -> (role_A, 6) : A.{IDd.IDc.Td.{Nd}_Kda}_Kd.{Tb.Mac1}_Kb
(role_A, 6) -> (role_B, 5) : B.{Tickt}_Kda.H(Tickt.Nd)
(role_B, 5) -> (role_C, 4) : C.{Tickt}_Kda.H(Tickt.Nd)
(role_C, 4) -> (role_D, 3) : D.{Tickt}_Kda.H(Tickt.Nd)
```

Figure 5.4: EAKES6Lo : AKE Phase

Step 1: 6LH → 6LR : $\{ID_{6LH}, ID_{6LR}, T_{6LH}, \{N_6LH)K_{6LH\_SER}\}\ K_{pri6LH}.\}$

Step 2: 6LR → 6LER: $\{ID_{6LH}, ID_{6LR}, T_{6LH}, \{N_6LH)K_{6LH\_SER}\}\ K_{pri6LH}.\}$

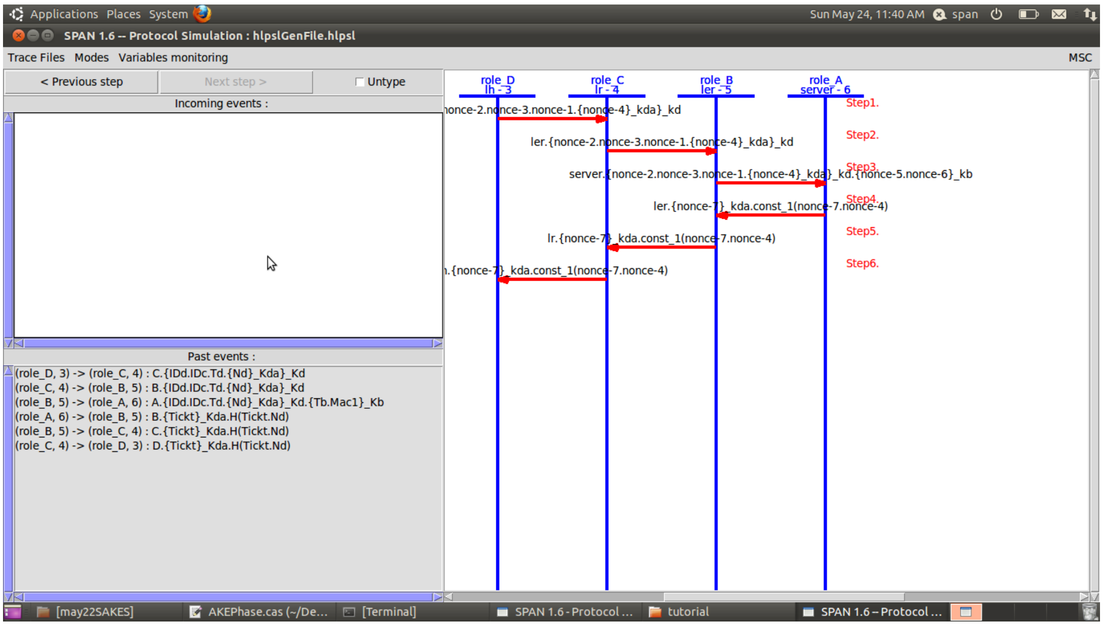
Step 3: 6LER → Server: $\{ID_{6LH},\ ID_{6LR},\ T_{6LH},\ \{N_{6LH}\}\ K_{6LH\_SER}\}K_{pri6LH},$ $\{T_{6LER},\ \mathrm{MAC1}\}\ K_{pri6LER}$

Step 4: Server → 6LER: $\{ID_{6LH},\ IPv6_{6LH},\ K_{pub6LH},\ T_{SER},\ T_{EXP}\ \}\ K_{priSER},$ MAC2

Step 5: 6LER → 6LR: $\{ID_{6LH},\ IPv6_{6LH},\ K_{pub6LH},\ T_{SER},\ T_{EXP}\ \}\ K_{priSER},$ MAC2

Step 6: 6LR → 6LH: $\{ID_{6LH},\ IPv6_{6LH},\ K_{pub6LH},\ T_{SER},\ T_{EXP}\ \}\ K_{priSER},$ MAC2

3. *Handover Phase*



```
nonce-2.nonce-3.nonce-1.{nonce-4}_kda}_kd
ler.{nonce-2.nonce-3.nonce-1.{nonce-4}_kda}_kd
server.{nonce-2.nonce-3.nonce-1.{nonce-4}_kda}_kd.{nonce-5.nonce-6}_kb
ler.{nonce-7}_kda.const_1(nonce-7.nonce-4)
lr.{nonce-7}_kda.const_1(nonce-7.nonce-4)
.{nonce-7}_kda.const_1(nonce-7.nonce-4)
```

```
(role_D, 3) -> (role_C, 4) : C.{IDd.IDc.Td.{Nd}_Kda}_Kd
(role_C, 4) -> (role_B, 5) : B.{IDd.IDc.Td.{Nd}_Kda}_Kd
(role_B, 5) -> (role_A, 6) : A.{IDd.IDc.Td.{Nd}_Kda}_Kd.{Tb.Mac1}_Kb
(role_A, 6) -> (role_B, 5) : B.{Tickt}_Kda.H(Tickt.Nd)
(role_B, 5) -> (role_C, 4) : C.{Tickt}_Kda.H(Tickt.Nd)
(role_C, 4) -> (role_D, 3) : D.{Tickt}_Kda.H(Tickt.Nd)
```

Figure 5.5: EAKES6Lo : Handover Phase

Step 1: 6LH → 6LR2: (Ticket,T_6LH, MAC3)

Step 2: 6LR2 → Server: (Ticket,T_6LH, MAC3)

Step 3: Server → 6LR1: $(ID_6LH,\ T_{Ser})\ K_{6LR1-Serv}$

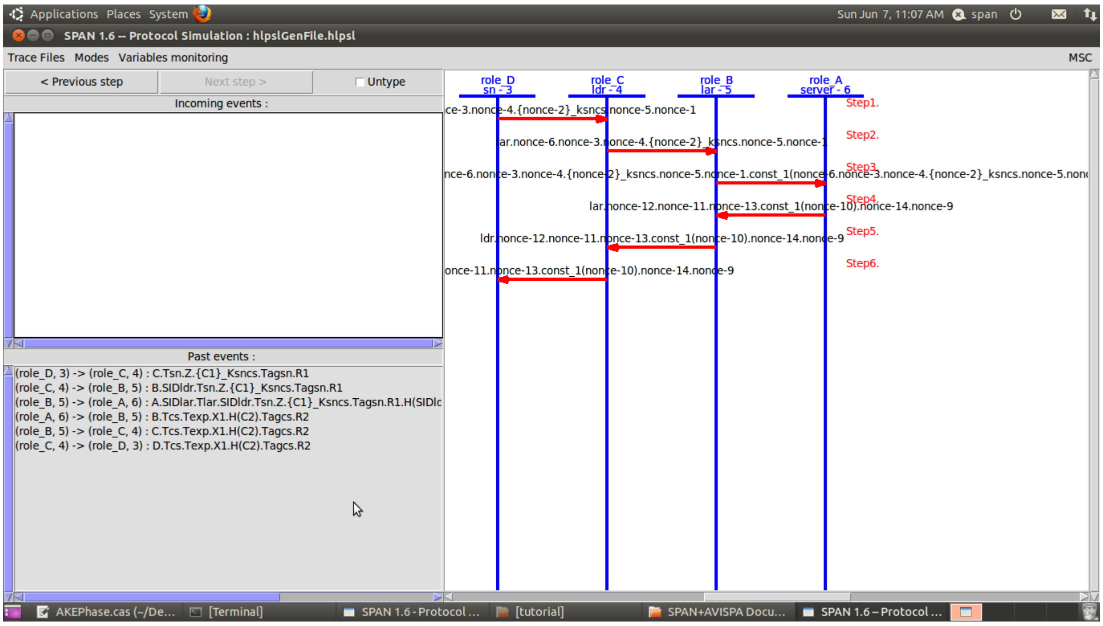
Step 4: 6LR1 → Server : ACK(T_{6LR1})

Step 5: Server → 6LR2 : $\{ID_{6LH},\ ID_{6LR2},\ T_{SER},\ \text{MAC4}\}\ K_{priSER},\ \{N_s\}\ K_{6LR2\_SER},\ \{N_s\}\ K_{6LH\_SER}$

Step 6: 6LR2 → 6LH: $\{ID_{6LH},\ ID_{6LR2},\ T_{SER},\ \text{MAC4}\}\ K_{priSER},\ \{N_s\}\ K_{6LH\_SER}$

5.1.3 S6AE

1. *AKE Phase*



```
ce-3.nonce-4.{nonce-2}_ksncs.nonce-5.nonce-1
lar.nonce-6.nonce-3.nonce-4.{nonce-2}_ksncs.nonce-5.nonce-1
nce-6.nonce-3.nonce-4.{nonce-2}_ksncs.nonce-5.nonce-1.const_1(nonce-6.nonce-3.nonce-4.{nonce-2}_ksncs.nonce-5.nonc
lar.nonce-12.nonce-11.nonce-13.const_1(nonce-10).nonce-14.nonce-9
ldr.nonce-12.nonce-11.nonce-13.const_1(nonce-10).nonce-14.nonce-9
once-11.nonce-13.const_1(nonce-10).nonce-14.nonce-9
```

```
(role_D, 3) -> (role_C, 4) : C.Tsn.Z.{C1}_Ksncs.Tagsn.R1
(role_C, 4) -> (role_B, 5) : B.SIDldr.Tsn.Z.{C1}_Ksncs.Tagsn.R1
(role_B, 5) -> (role_A, 6) : A.SIDlar.Tlar.SIDldr.Tsn.Z.{C1}_Ksncs.Tagsn.R1.H(SIDlc
(role_A, 6) -> (role_B, 5) : B.Tcs.Texp.X1.H(C2).Tagcs.R2
(role_B, 5) -> (role_C, 4) : C.Tcs.Texp.X1.H(C2).Tagcs.R2
(role_C, 4) -> (role_D, 3) : D.Tcs.Texp.X1.H(C2).Tagcs.R2
```

Figure 5.6: S6AE : AKE Phase

Step 1: SN → 6LDR : M1 : $\langle T_{sn} \parallel \mathrm{Z} \parallel \langle C_1 \parallel Tag_{sn} \rangle \parallel R_1 \rangle$

Step 2: 6LDR → 6LAR: M2 : $\langle SID_{ldr} \parallel \mathrm{M1} \rangle$

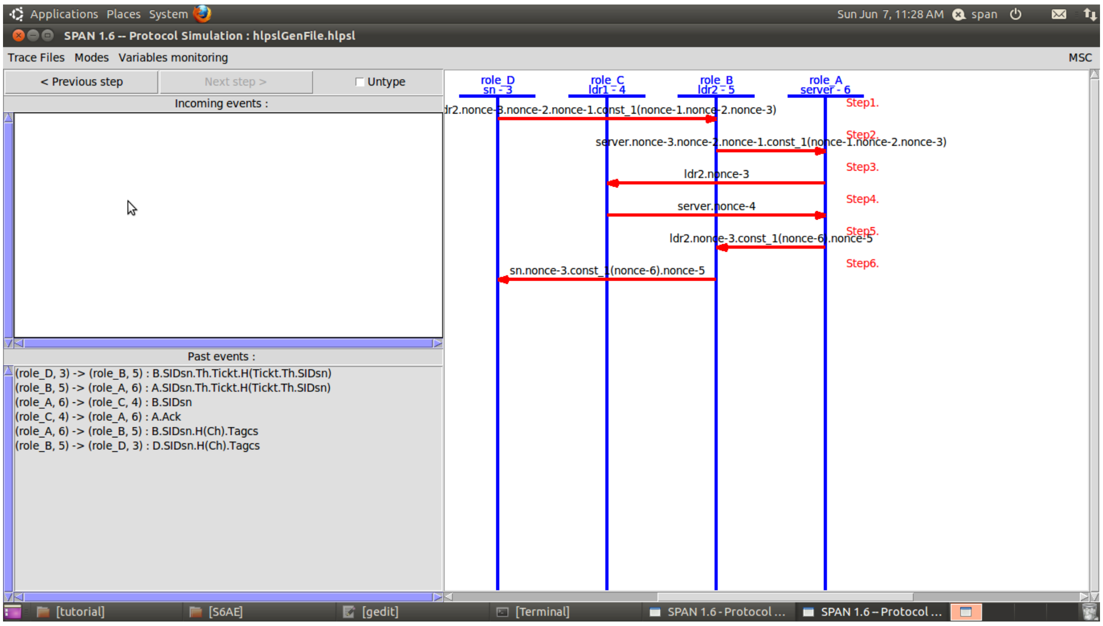
Step 3: 6LAR → CS: M3: $\langle SID_{lar} \parallel T_{lar} \parallel \mathrm{M2} \parallel H_{lar} \rangle$

Step 4: CS → 6LAR: M4: $\langle T_{cs} \parallel T_{exp} \parallel \mathrm{X1} \parallel \langle C_2 \parallel Tag_{cs} \rangle \parallel R_2 \rangle$

Step 5: 6LAR → 6LDR: Relay M4

Step 6: 6LDR → SN: Relay M4

2. *Handover Phase*



```
ldr2.nonce-3.nonce-2.nonce-1.const_1(nonce-1.nonce-2.nonce-3)
server.nonce-3.nonce-2.nonce-1.const_1(nonce-1.nonce-2.nonce-3)
ldr2.nonce-3
server.nonce-4
ldr2.nonce-3.const_1(nonce-6).nonce-5
sn.nonce-3.const_1(nonce-6).nonce-5
```

```
(role_D, 3) -> (role_B, 5) : B.SIDsn.Th.Tickt.H(Tickt.Th.SIDsn)
(role_B, 5) -> (role_A, 6) : A.SIDsn.Th.Tickt.H(Tickt.Th.SIDsn)
(role_A, 6) -> (role_C, 4) : B.SIDsn
(role_C, 4) -> (role_A, 6) : A.Ack
(role_A, 6) -> (role_B, 5) : B.SIDsn.H(Ch).Tagcs
(role_B, 5) -> (role_D, 3) : D.SIDsn.H(Ch).Tagcs
```

Figure 5.7: S6AE : Handover Phase

Step 1: SN → 6LDR2: M_{h1}: $\langle SID_{sn} \parallel T_h \parallel T_{ic}^{sn} \parallel H_h \rangle$

Step 2: 6LDR2 → CS: Forwards M_{h1}

Step 3: CS → 6LDR1: ID_{sn}

Step 4: 6LDR1 → CS: ID_{sn}, ACK

Step 5: CS → 6LDR2: M_{h2}: $\langle SID_{sn} \parallel C_h \parallel Tag_{cs} \rangle$

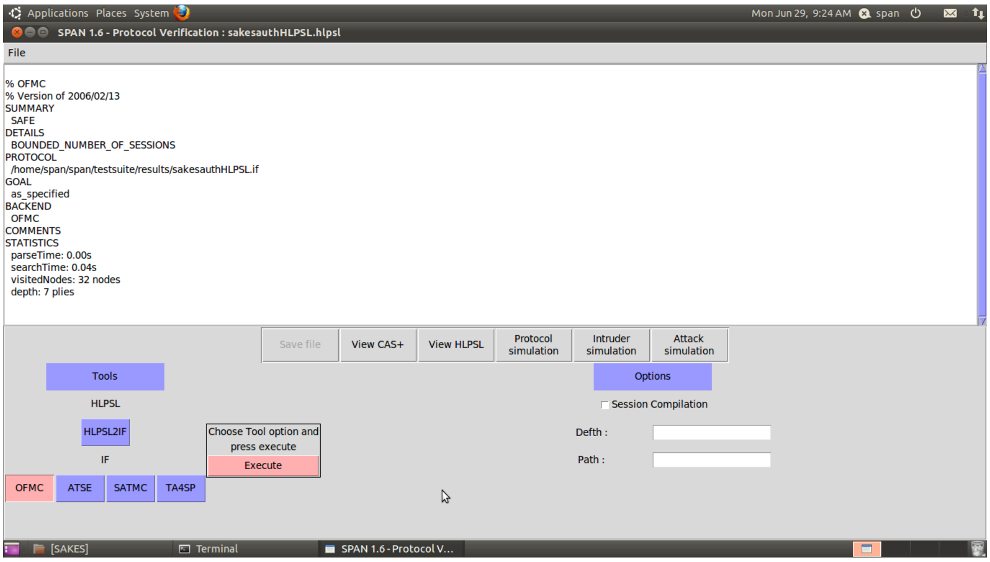
Step 6: 6LDR2 → SN: Forwards M_{h2}

5.2 Crypt-Analysis Using AVISPA

Crypt-analysis of the following schemes is conducted using the AVISPA tool, which obeys the DY attack model and is commonly used by the research community to examine the capabilities of the security algorithms. AVISPA comprises four back-end models, known as CL-AtSe, TA4SP, OFMC, and SATMC. These back-ends perform various automatic analyses to detect vulnerabilities in the security scheme. The back-end chose for the analysis of the following schemes is OFMC. It uses perfect cryptography, which means that the adversary cannot derive the messages or plaintext from ciphertext without perceiving the secret key.

5.2.1 SAKES

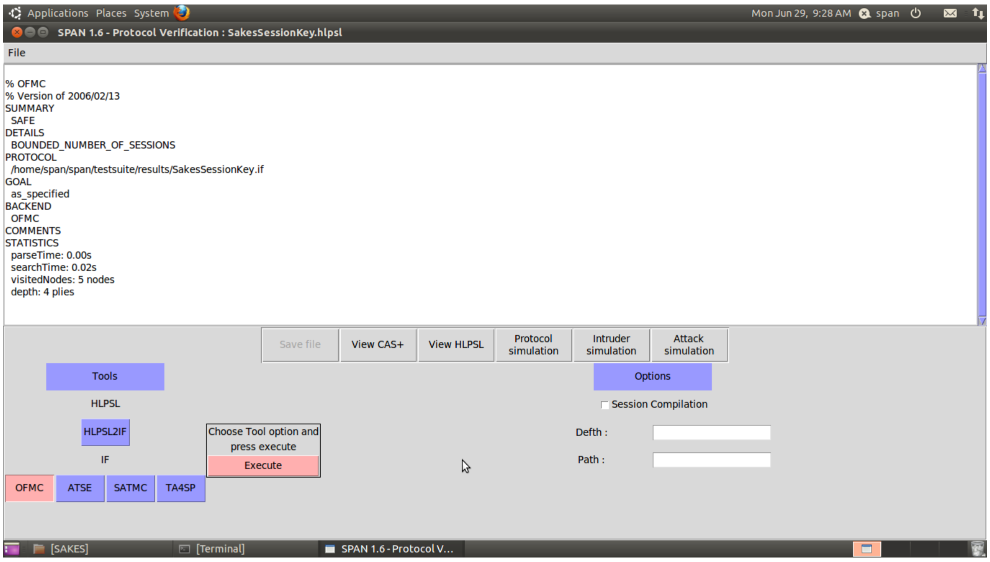
1. Authentication Phase



```
% OFMC
% Version of 2006/02/13
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
 /home/span/span/testsuite/results/sakesauthHLPSL.if
GOAL
 as_specified
BACKEND
 OFMC
COMMENTS
STATISTICS
 parseTime: 0.00s
 searchTime: 0.04s
 visitedNodes: 32 nodes
 depth: 7 plies
```

Figure 5.8: SAKES : Authentication Phase - OFMC Back-end

2. Key Establishment Phase

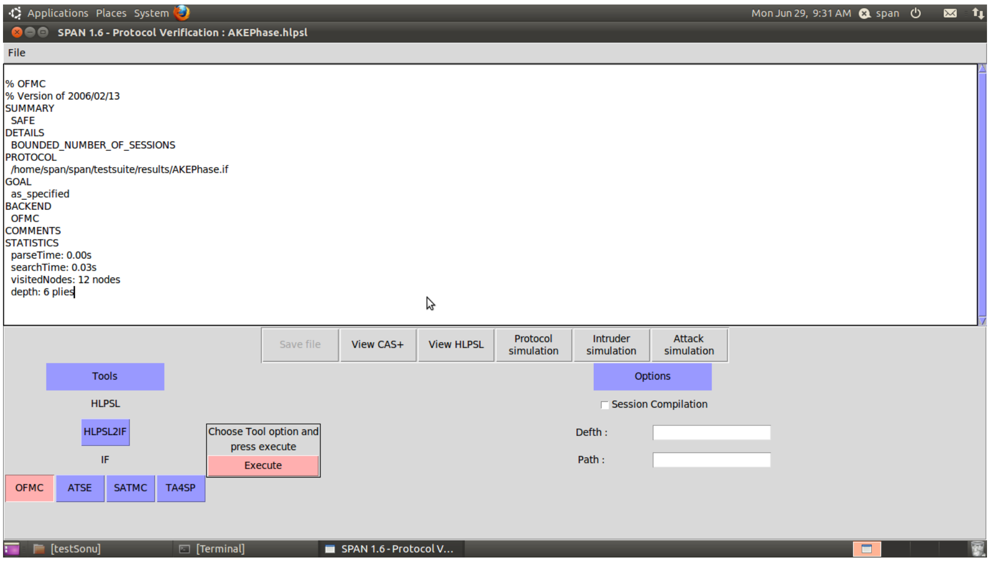


```
% OFMC
% Version of 2006/02/13
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
 /home/span/span/testsuite/results/SakesSessionKey.if
GOAL
 as_specified
BACKEND
 OFMC
COMMENTS
STATISTICS
 parseTime: 0.00s
 searchTime: 0.02s
 visitedNodes: 5 nodes
 depth: 4 plies
```

Figure 5.9: SAKES : Key Establishment Phase - OFMC Back-end

5.2.2 EAKES6Lo

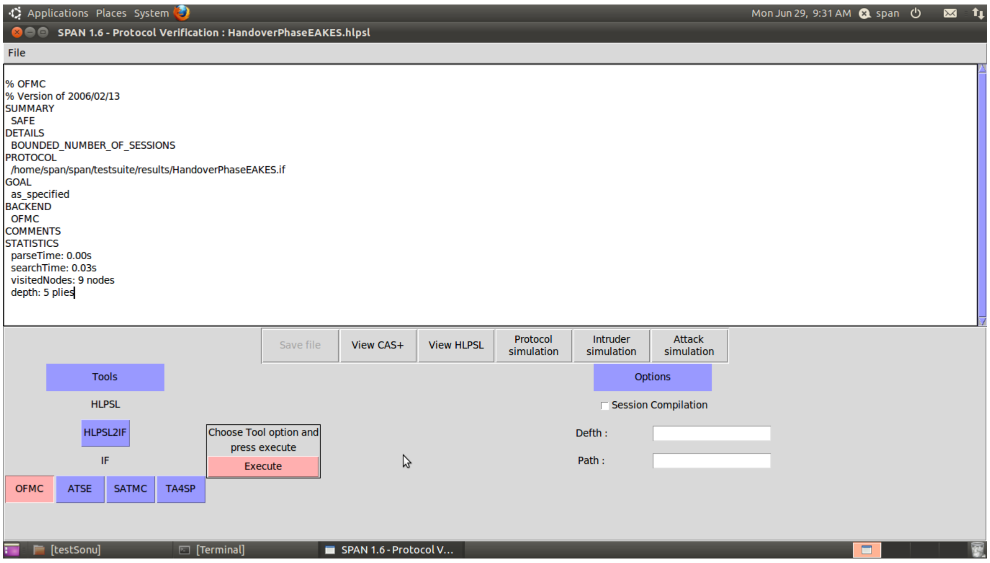
1. AKE Phase



```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/AKEPhase.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.03s
  visitedNodes: 12 nodes
  depth: 6 plies
```

Figure 5.10: EAKES6Lo : AKE Phase - OFMC Back-end

2. Handover Phase

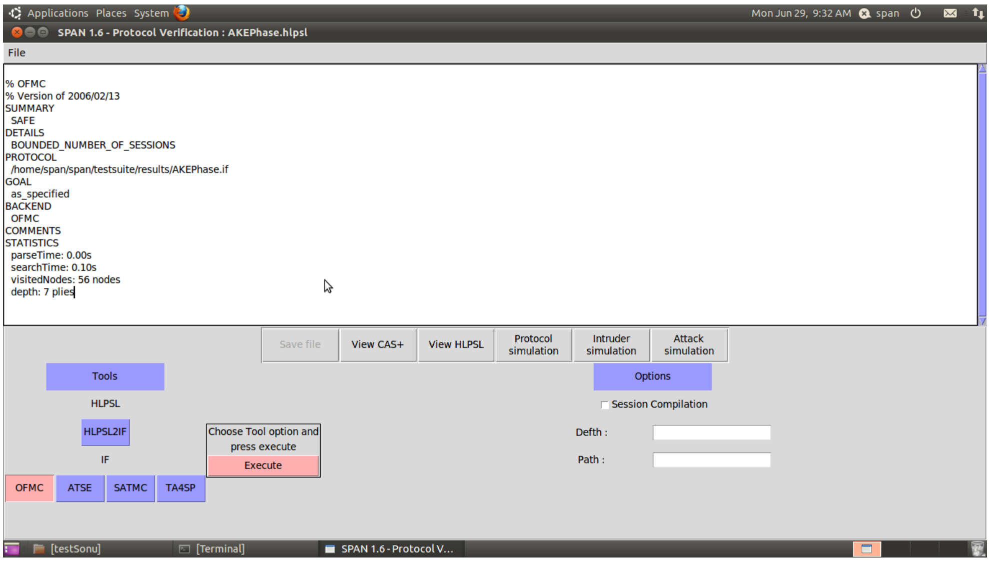


```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/HandoverPhaseEAKES.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.03s
  visitedNodes: 9 nodes
  depth: 5 plies
```

Figure 5.11: EAKES6Lo : Handover Phase - OFMC Back-end

5.2.3 S6AE

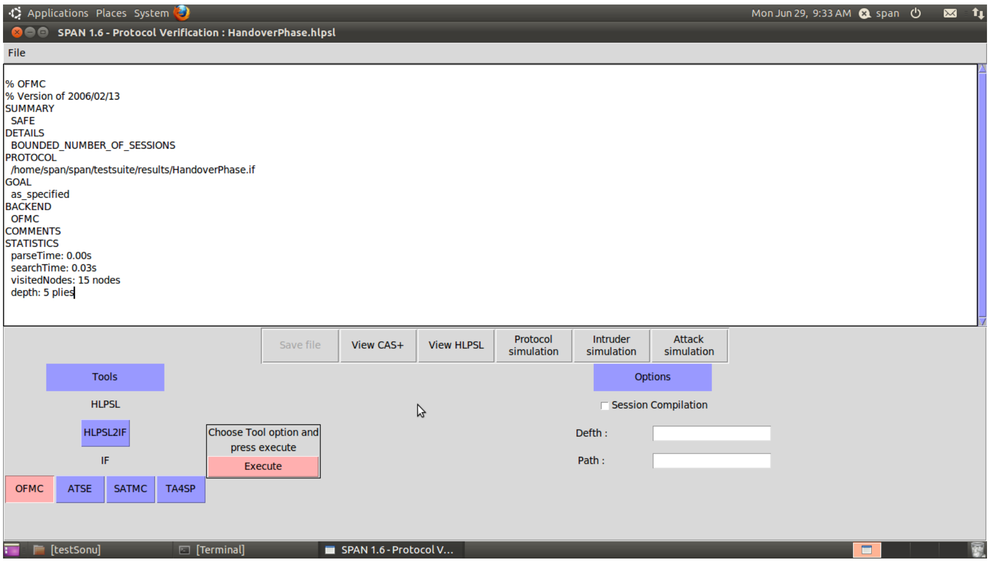
1. AKE Phase



```
% OFMC
% Version of 2006/02/13
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
 /home/span/span/testsuite/results/AKEPhase.if
GOAL
 as_specified
BACKEND
 OFMC
COMMENTS
STATISTICS
 parseTime: 0.00s
 searchTime: 0.10s
 visitedNodes: 56 nodes
 depth: 7 plies
```

Figure 5.12: S6AE : AKE Phase - OFMC Back-end

2. Handover Phase



```
% OFMC
% Version of 2006/02/13
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
 /home/span/span/testsuite/results/HandoverPhase.if
GOAL
 as_specified
BACKEND
 OFMC
COMMENTS
STATISTICS
 parseTime: 0.00s
 searchTime: 0.03s
 visitedNodes: 15 nodes
 depth: 5 plies
```

Figure 5.13: S6AE : Handover Phase - OFMC Back-end

5.3 Analysis

Table 5.1: Notation used in Analysis

Notation	Description
AES-CTR-128 bits	AES (Advanced Encryption Standard) is one of the most used algorithms for block encryption. AES-CTR is one of the five modes of AES known as counter mode.
ECDSA-160	ECDSA stands for "Elliptic Curve Digital Signature Algorithm", it's used to create a digital signature of data (a file for example) in order to allow you to verify its authenticity without compromising its security.
CS,SN	Central Server and 6LoWPAN Sensor node
6LDR	6LoWPAN Domain Router
6LAR	6LoWPAN Access Router
SID_x	Pseudo-identity of x
ID_x	Secret real identity of x
SP_1, SP_1^n	Secret parameter used in authentication process
T_{aes}	The average time required for AES-128
$T_{exponent}$	The average time required for the modular exponentiation (DiffieHellman)
T_{sha}	The average time required for SHA-256
T_{sg}	The average time required for ECC public/private key generation
T_{sv}	The average time needed for the signature generation/verification
T_{ascon}	The average average time required for ASCON
T_{xor}	The average average time required for XOR
T_{ic}^{sn}	Ticket shared between sensor node and central server in S6AE scheme
T_{exp}	Expiry time of the ticket in S6AE scheme
$Nonce_X$	Nonce generated by X
$SecretKey_{XY}$	Secret key shared between X and Y
T_{Eakes}	Tickets issued for handover phase in EAKES6Lo scheme
ID_x	Identity of X
MAC_x	MAC address of x

5.3.1 Performance Analysis

1. *Type of Cryptographic System used for key exchange process*

   SAKES: Hybrid security scheme and applies the Diffie-Hellman (DH) key exchange mechanism

   EAKES6Lo: AES-CTR-128 bits, SHA-256, and ECDSA-160

S6AE: SHA-256 and ASCON

2. *Storage Overhead* (Parameters required to be stored by SN and CS)

   SAKES: SN is required to store $\{ID_6LDR, Nonce_{6LAR}, SecretKey_{CS}, Nonce_{6LDR}\}$ (64+32+2048+32)=2176 bits. Total = 272 bytes. CS needs to store the parameters $\{ID_6LDR, ID_{SN}, SecretKey_{SN}\}$ (64+64+2048)=2176 bits. Total = 272 bytes.

   EAKES6Lo: SN is reqd. to store $\{ID_{SN}, ID_{6LDR}, Key_{SN-Serv}, Nonce_{SN}, Nonce_{Server}, T_{Eakes}\}$ (64+64+256+32+32+256) = 704 bits. Total = 88 bytes. CS needs to store the parameters $\{ID_{6SN}, SecretKey_{Server-SN}, Nonce_{SN}, Nonce_{Server}, T_{Eakes}\}$ (64+256+32+32+256) = 640 bits.Total = 80 bytes.

   S6AE: SN is required to store the tuple $\{ID_{sn}, SID_{sn}, SP_1^n, T_{ic}^{sn}, MAC_{cs}, T_{exp}\}$, which requires (64 + 64 + 64 + 128 + 48 + 32) = 400 bits. Total in SN side : 50 bytes. CS needs to store the parameters $\{SID_{sn}, ID_{sn}, SP_1, SP_1^n, T_{ic}^{sn}, MAC_{sn}, T_{exp}\}$, which requires (64 + 64 + 64 + 64 + 128 + 48 + 32) = 464 bits. Total in CS side : 58 bytes.

3. *Computational Overhead*

   SAKES:

   SN: $3T_{aes}+ T_{sha}$, 6LDR: $2T_{exponent} + 2T_{aes} + T_{sha}$, 6LAR: n/a, CS: $T_{exponent}+ 3T_{aes}+ 2T_{sha}$

   Total time: $3T_{exponent}+ 8T_{aes}+ 4T_{sha}$

   $\approx$ 58.6044 ms

   EAKES6LO:

   SN: $2T_{aes}+ T_{sha} + T_{sg}$, 6LDR: T_{sv}, 6LAR: $T_{sha} + {}_{aes}$, CS: $3T_{aes}+ 2T_{sha}+ {}_{sv}$

   Total time: $5T_{aes}+ 4T_{sha} + 2_{sv} + T_{sg}$

   $\approx$ 17.2494 ms

   S6AE:

   SN: $2T_{ascon}+ 5T_{sha} + 7T_{xor}$, 6LDR: T_{xor}, 6LAR: T_{sha}, CS: $2T_{ascon} + 7T_{sha}+ 14T_{xor}$

   Total time: $4T_{ascon}+ 13T_{sha} + 22T_{xor}$

≈ 0.6643 ms

4. *Handover Phase*

   SAKES:

   It doesn't support mobile nodes.

   EAKES6Lo:

   $6T_{aes} + T_{sg} + T_{sv} + 6T_{sha}$

   Time required: 11.9366 ms

   S6AE:

   $2T_{ascon} + 4T_{sha}$

   Time required: 0.2544 ms

5. *Communication Overhead*

   SAKES

   SN → 6LDR: 688 bits

   6LDR → SN: 2176 bits

   Total: 2864 bits = 358 bytes

   EAKES6Lo

   SN → 6LDR: 672 bits

   6LDR → SN: 784 bits

   Total: 1456 bits = 182 bytes

   S6AE

   SN → 6LDR: 496 bits

   6LDR → SN: 528 bits

   Total: 1024 bits = 128 bytes

5.3.2 Security Analysis

Table 5.2: Security Comparison

	SAKES	EAKES6Lo	S6AE
1. Replay Attack	✓	✓	✓
2. MITM	✓	✓	✓
3. Impersonation Attack	✓	✓	✓
4. Sybil Attack	✓	✓	✓
5. Compromised Attack	x	✓	✓
6. IP Spoofing Attack	x	x	✓
7. Identity Preservation Attack	x	x	✓
8. Jamming Attack	x	x	x

Chapter 6

Conclusion and Future Work

6LoWPAN is a providential technology having a vital share in IoT and is commonly deployed in a variety of applications. Originally, 6LoWPAN does not provide any security and privacy mechanism. In this project work the schemes for authentication and key establishment in 6LoWPAN have been studied and analyzed in details. The schemes have been analyzed based on the factors: type of cryptographic system used for key exchange process, computational overhead, communication overhead, storage overhead and time required in handover phase. From the analysis, it is found that the S6AE scheme is efficient and takes up less resource as compared to SAKES and EAKES6Lo which is better for resource constrained devices in 6LoWPAN network. The security verification using AVISPA illustrates that the schemes are secure against various malicious attacks. However, all the schemes are found to be vulnerable to jamming attack.

As a future work, the security gaps which cannot be overcome by the available schemes can be worked on to improve existing schemes or develop a new solution. The existing schemes SAKES could be improved to introduce mobile nodes, EAKES6Lo could be improved to be less computationally expensive and S6AE can be extended to varying security levels using secure cryptographic algorithms.

Bibliography

[1] J. Kim, J. Lee, J. K. and Yun, J. M2m service platforms: Survey, issues, and enabling technologies. *IEEE Commun. Surveys. Tuts.* **16**(1), 61–67 (2014).

[2] Gomes, T.; Salgado, F. P. S. C. J. T. A. A 6lowpan accelerator for internet of things endpoint devices. *IEEE Internet Things J* **5**, 371–377 (2017).

[3] Gomez, C.; Paradells, J. B. C. C. J. From 6lowpan to 6lo: Expanding the universe of ipv6-supported technologies for the internet of things. *IEEE Commun. Mag* **55**, 148–155 (2017).

[4] Hennebert, C.; Santos, J. Security protocols and privacy issues into 6lowpan stack: A synthesis. *IEEE Internet Things J* **1**, 384–398 (2014).

[5] Kushalnagar, N.; Montenegro, G. Transmission of ipv6 packets over ieee 802.15.4 networks. *IEEE Commun. Mag* **4944**, 130 (2007).

[6] Ishaq, I. Carels, D. T. G. H. J. A. F. P. E. M. I. D. P. Ietf standardization in the field of the internet of things (iot): A survey. *J. Sens. Actuator Netw* **2**, 235–287 (2013).

[7] Butun, I.; Österberg, P. S. H. Security of the internet of things: Vulnerabilities, attacks and countermeasures. *IEEE Commun. Surv. Tutor.* **22**, 616–644 (2019).

[8] ElectronicNotes. What is 6lowpan - the basics (https://www.electronics-notes.com/articles/connectivity/ieee-802-15-4-wireless/6lowpan.php).

[9] Dolev, D.; Yao, A. On the security of public key protocols. *Trans. Inform. Theory* **29**, 198–203 (1983).

[10] Commercial national security algorithm suite and quantum computing faq. *U.S. National Security Agency* **29**, 198–203 (January, 2016).

[11] OmniSecu.com. Ipv6 ndp (neighbour discovery protocol) and important functions of ipv6 ndp (https://www.omnisecu.com/tcpip/ipv6/ndp-neighbour-discovery-protocol-functions-of-ndp.php).

[12] P. Kasinathan, C. Pastrone, M. A. S. and Vinkovits, M. Denial-ofservice detection in 6lowpan based internet of things. *IEEE 9th Int. Conf. Wireless Mobile Comput., Netw. Commun.* , 600–607 (2013).

[13] Raza, S.; Seitz, L. S. D. S. G. S3k: Scalable security with symmetric keys-dtls keynestablishment for the internet of things. *IEEE Trans. Autom. Sci. Eng.* **13**, 1270–1280 (2016).

[14] Glissa, G.; Meddeb, A. 6lowpsec: An end-to-end security protocol for 6lowpan. *Ad Hoc Netw* **82**, 100–112 (2019).

[15] Hussen, H.R.; Tizazu, G. T. M. L. T. C. Y. K. K. Sakes: Secure authentication and key establishment scheme for m2m communication in the ip-based wireless sensor network (6lowpan). *2013 Fifth International Conference on Ubiquitous and Future Networks (ICUFN)* , 246–251 (2013).

[16] Qiu, Y.; Ma, M. A mutual authentication and key establishment scheme for m2m communication in 6lowpan networks. *IEEE Trans. Ind. Inform* **12**, 2074–2085 (2016).

[17] Muhammad Tanveer, Ghulam Abbas, Z. H. A. M. W. F. M. and Kim, S. S6ae: Securing 6lowpan using authenticated encryption scheme. *ncbi.nlm.nih.gov* (2020).