UNIVERSITY of LIMERICK

OLLSCOIL LUIMNIGH

Department of Electronic & Computer Engineering

Logic-based Verification of Security Protocols

Shireen Kudukkil Manchingal

ID: 16073355

Master of Engineering Information and Network Security

This project was undertaken under the supervision and direction of

Dr Reiner Dojen

Department of Electronic & Computer Engineering

University of Limerick

Submitted to the University of Limerick

September 2017

Table of Contents

CHAPTER 1: INTRODUCTION 1

1.1 PROJECT OBJECTIVES 2

1.2 REPORT OUTLINE 3

CHAPTER 2: SECURITY PROTOCOLS 4

2.1 OBJECTIVES OF SECURITY PROTOCOLS 4

2.2 SECURITY IN PROTOCOL DESIGN 5

2.3 ATTACKS ON SECURITY PROTOCOLS 7

2.3.1 Replay attacks 8

2.3.2 Parallel Session Attacks 8

CHAPTER 3: SECURITY PROTOCOLS FOR MOBILE COMMUNICATION 10

3.1 CLASSIQUANTUM RESISTANCE SECURE SIMPLE PAIRING (CRSSP) - 2017 11

3.2 HIP-BASED SECURE SOFTWARE DEFINED MOBILE NETWORKS - 2017 14

3.2.1 HIP-based control channel 16

3.2.2 HIP-based data channel 19

3.3 SECURE MOBILE WALLET – 2017 20

3.4 SECURE PATIENT-HEALTH MONITORING WIRELESS BODY AREA NETWORKS - 2016 23

3.5 P2P SERVICE SECURITY PROTOCOL IN M2M ENVIRONMENT - 2016 26

3.6 SECURE COMMUNICATION IN VEHICULAR NETWORKS - 2016 27

3.7 TTP BASED HIGH-EFFICIENT MULTI-KEY EXCHANGE PROTOCOL (THMEP) - 2016 28

3.8 MAKA SCHEME WITH USER ANONYMITY IN GLOMONET – 2015 30

CHAPTER 4: VERIFICATION OF SECURITY PROTOCOLS 32

4.1 STATE MACHINE-BASED TECHNIQUE 33

4.2 ALGEBRAIC TERM REWRITING TECHNIQUES 33

4.3 THEOREM PROVING TECHNIQUES 33

4.4 MODAL LOGICS 34

CHAPTER 5: LOGIC-BASED VERIFICATION **35**

5.1 AUTOMATION OF SECURITY PROTOCOL VERIFICATION 36

CHAPTER 6: CRYPTOGRAPHIC-PROTOCOL DESIGN & VERIFICATION TOOL ... **38**

6.1 INPUT LANGUAGE 40

CHAPTER 7: LOGIC-BASED VERIFICATION OF SECURITY PROTOCOLS **43**

7.1 PROJECT DESCRIPTION 44

7.2 VERIFICATION OF SAMPLE PROTOCOLS 45

7.2.1 Formal Verification of Kao Chow Authentication v.1 ..... *45*

7.2.2 Formal Verification of Andrew Secure RPC protocol *51*

7.2.3 Formal Verification of Needham-Schroeder Public Key protocol *52*

7.2.4 Formal Verification of Beller-Chang-Yacobi protocol *54*

7.3 CONCLUSION 55

CHAPTER 8: LOGIC-BASED VERIFICATION OF SECURITY PROTOCOLS FOR MOBILE/WIRELESS COMMUNICATION USING CDVT **56**

8.1 INTRODUCTION 56

8.2 CLASSIQUANTUM RESISTANCE SECURE SIMPLE PAIRING (CRSSP) 56

8.2.1 Formal Verification of CRSSP protocol *58*

8.2.2 Verification results of CRSSP in CDVT *63*

8.3 HIP-BASED SECURE SOFTWARE DEFINED MOBILE NETWORKS (SDMN) 64

8.3.1 HIP-based control channel *64*

8.3.2 Formal Verification of HIP-based control channel *65*

8.3.3 Verification results of HIP-based control channel in CDVT *68*

8.3.4 HIP-based data channel *69*

8.3.5 Formal Verification of HIP-based data channel *70*

8.3.6 Verification results of HIP-based data channel in CDVT *73*

8.4 SECURE MOBILE WALLET 75

8.4.1 Formal Verification of Secure Mobile Wallet *76*

8.4.2 Verification results of secure mobile wallet in CDVT *81*

8.5 SECURE PATIENT-HEALTH MONITORING WIRELESS BODY AREA NETWORKS 83
8.5.1 Formal Verification of patient-health monitoring protocol in WBAN........................84
8.5.2 Verification results of patient-health monitoring WBAN in CDVT 90
8.6 P2P SERVICE SECURITY PROTOCOL IN M2M ENVIRONMENT 92
8.6.1 Formal Verification of P2P service security protocol..93
8.6.2 Verification results of P2P service security protocol in CDVT..................................97
8.7 SECURE COMMUNICATION IN VEHICULAR NETWORKS 98
8.7.1 Formal Verification of secure communication protocol in vehicular network..........100
8.7.2 Verification results of secure vehicular communication protocol in CDVT.............102
8.8 TTP BASED HIGH-EFFICIENT MULTI-KEY EXCHANGE PROTOCOL (THMEP)....................103
8.8.1 Formal Verification of THMEP protocol 104
8.8.2 Verification results of THMEP protocol in CDVT 109
8.9 MAKA SCHEME WITH USER ANONYMITY IN GLOMONET...111
8.9.1 Formal Verification of MAKA scheme...112
8.9.2 Verification results of MAKA scheme in CDVT...117

CHAPTER 9: ACTION PLAN 119

CHAPTER 10: CONCLUSION...122

REFERENCES 123

APPENDIX A - SAMPLE PROTOCOL SPECIFICATIONS 129

Table of Figures

Figure 1 : Exchanging session keys [6]..6
Figure 2: Working of CRSSP protocol 13
Figure 3: HIP-based secure SDMN communication channel architecture 14
Figure 4: Tunnel establishment procedure for HIP-based control channel [13]...........................18
Figure 5: Tunnel establishment procedure for HIP-based data channel 20

Figure 6: Working of mobile wallet 21
Figure 7: WBAN architecture 24
Figure 8: Logic-based verification 36
Figure 9: The CDVT verification tool [23] 39
Figure 10: Logic-based verification of security protocols 43
Figure 11: Kao Chow Authentication v.1 46
Figure 12: Verification Results for Kao Chow v.1 50
Figure 13: Verification Result of CRSSP in CDVT 63
Figure 14: Verification result of HIP-based control channel in CDVT 69
Figure 15: Verification result of HIP-based data channel in CDVT 74
Figure 16: Verification Result of Secure Mobile Wallet in CDVT 82
Figure 17: Verification result of patient-health monitoring WBAN in CDVT 91
Figure 18: Verification result of P2P service security protocol in CDVT 98
Figure 19: Verification result of secure vehicular network communication protocol in CDVT .103
Figure 20: Verification result of THMEP protocol in CDVT 110
Figure 21: Verification result of MAKA scheme in CDVT 118
Figure 22: Gantt Chart for Autumn and Spring 120
Figure 23: Gantt Chart for Summer 121

Table of Tables

Table 1: Atomic units of textual grammar 41
Table 2: Composite data construction 42
Table 3: Statement construction 42

Abstract

Recent technologies have cleared the course for large scale application of electronic communication. The open and distributed nature of these communications implies that the communication medium is no longer fully controlled by the communicating parties. As a result, there has been an increasing demand for research in establishing secure communications over insecure networks. Data security is as extremely important aspect and it has become more technically complex than in the past. Authentication, encryption, integrity and safety are necessary in promoting data security. Various security protocols are employed so as to implement security over networks and prevent unauthorized access, thereby achieving secure communication between entities.

The increasing use of mobile communication networks results in ever more stringent security requirements. Mobile communication is more susceptible to security attacks like unauthorized access, loss of data, eavesdropping and interception than fixed network communications. Therefore, solutions to secure mobile communications are necessary in guaranteeing privacy and authentication of users.

This thesis presents the automated formal verification of eight recent security protocols for wireless/mobile communication using the Crytpographic procotol Development and Verification Tool (CDVT). The verification results of these protocols are analyzed to detect any weaknesses or vulnerabilities in the protocol and to determine the accuracy of the protocol. These weaknesses make the protocol vulnerable to a range of attacks such as replay attacks, man-in-the-middle attacks, parallel session key attacks, known key attacks, eavesdropping, freshness attacks, etc. This project also provides insight on how these protocols are specified using the CDVT language and the limitations of the same.

.

Chapter 1: Introduction

Security protocols, also called cryptographic protocols, outline the method of exchanging information securely among the communicating entities in a network. They are abstract protocols that describe how an algorithm must be defined; it utilizes cryptographic approaches and performs security-related functions to maintain information authenticity. Security protocols use cryptographic techniques to attain goals like message integrity, confidentiality, service authentication, non-repudiation, order of messages, and distribution of cryptographic keys. With the absence of such authentication and security, a malicious attacker would impersonate any of the communication parties and obtain unauthorized access to the network.

Unfortunately, distributed systems and open networks are often susceptible to hostile intruders (adversaries) who may try and subvert the protocol design goals. Existing flaws within the protocol design may be maliciously employed by an adversary to attack the protocol and compromise the security property of the protocol. There have been several examples of cryptographic and security-based protocols that were published, believed to be sound, and later shown to have several security flaws.

Formal verification of cryptographic security protocols has been in use since many years as it has proven to be one the best methods to verify security protocols. Logic-based verification is one among the most widely used formal verification techniques within the context of security protocols. This owes to both, the simplicity of logic-based approaches and the conciseness of the proof they generate. Logic-based formal verification of cryptographic protocols aims at proving the correctness or accuracy of the protocol against its goals.

Modal logic consists of various statements regarding belief in or knowledge about messages in a distributed system, as well as inference rules for the derivation of new beliefs from available beliefs and/or new knowledge from available knowledge. The BAN logic, developed by Burrow, Abadi and Needham [1], provided the initial impetus in applying modal logic in a proof-based environment. Ever since the development of BAN, several researchers have applied numerous logic-based approaches for the formal verification of security protocols. All of these logics are used successfully to identify a number of flaws in several protocols.

It is extremely difficult to manually perform logic-based verification as the application of postulates of a verification logic is tedious and error-prone. Hence, the automation of such a verification process can remove the potential errors in the manual process. Additionally, the automation process saves up on the time taken to perform verification.

The cryptographic-protocol design & verification tool (CDVT) is an automated verification engine that implements a verification logic using the concept of layered proving trees [2]. The CDVT tool uses a parser to read in the formalized protocol specification along with the protocol goals and initial assumptions as a text file. The tool then applies the postulates of the implemented logic and the outcome of the automated verification result allows the user to trace the performed proofs. Therefore, if the verification failed, the user can analyze the reason for protocol failure.

The inference rules provided in the CDVT are the standard rules of natural deduction [3]. The axioms of the logic of knowledge express the fundamental properties of public-key cryptographic protocols.

1.1 Project Objectives

The major objective of this project is to review and formally verify a range of security protocols using a system for automated logic-based protocol verification. This project will initially focus on cryptographic security protocols and their vulnerabilities with a detailed study on security attacks. A formal verification of the security protocols will then be performed using an automated verification engine called the cryptographic-protocol design & verification tool (CDVT) [4]. The verification of these protocols will reveal new flaws in the security protocol design. An attempt to rectify these flaws and amend the protocol design will be carried out.

This project will target a range of recently proposed security protocols for mobile communication.

As a conclusion, the objective of this project is to:

- Perform an empirical study on cryptographic security protocols, various attacks on security protocols and security protocols for mobile communication
- Familiarize with logic-based analysis and verification of security protocols

- Formally verify eight recent security protocols for wireless communication
- Study about automated logic-based verification engine, analyze cryptographic-protocol design & verification tool (CDVT) and verify security protocols using CDVT tool.

1.2 Report Outline

The rest of this report is structured as follows:

- Chapter 2 provides a detailed study on security protocols, their objectives, security approaches and various attacks on security protocols
- Chapter 3 provides information about eight recent security protocols used in mobile communications.
- Chapter 4 describes the process of verification of security protocols where formal verification techniques are elaborated
- Chapter 5 focuses on logic-based verification techniques
- Chapter 6 provides an overview about the CDVT verification engine and its working.
- Chapter 7 includes a discussion about the project logic-based verification of security protocols.
- Chapter 8 contains the formal verifications and results of eight recent security protocols for wireless communication using CDVT tool
- Chapter 9 describes the detailed action plan of the project with the help of a Gantt chart
- Chapter 10 includes the references used

Chapter 2: Security Protocols

Security has recently become a primary concern as communication and information have become key factors in economic and social development. Networks and information systems are currently supporting services and carrying data to an extent inconceivable only a couple of years ago. Issues concerning security of electronic networks and information systems are growing along with the rapid increase in the range of network users and the value of their transactions. Cryptographic security protocols are used to provide security in communication and information systems by offering services such as authentication, key establishment, confidentiality, integrity, service availability and non-repudiation.

Cryptographic protocols, also referred to as security protocols, employ cryptographic methods as sequences of cryptographic primitives to provide secure communication between hosts over a network.

Needham and Schroeder [5] first proposed cryptographic security protocols where they published two different protocols for interactive communication, one using shared key encryption and one using public key encryption. These protocols were the basis for the development of subsequent work, e.g., the Kerberos authentication protocol.

2.1 Objectives of Security Protocols

Security protocols are often required to satisfy certain traditional requirements such as confidentiality and authentication to be considered as secure. Most cryptographic security protocols implement the following objectives:

- **Authentication** is the process of verifying an identity claimed by or for a system entity, which may be a peer in a communication or the source of some data. The verification is achieved presenting authentication information or credentials that confirms the binding between the entity and the identifier, i.e., through certificates or digital signatures. Authentication is usually divided into *peer entity authentication* and *data origin authentication.* In peer entity authentication, the process attempts to verify that a peer entity in an association is as claimed. Data origin authentication provides the receiver with evidence regarding the identity of the sender.

- **Confidentiality** is the property that a data item is not disclosed to unauthorized individuals, entities, or processes, and remains unknown to the intruder. Hence, confidentiality is the assurance of *data privacy*.

- **Data integrity** refers to assurance of the accuracy and consistency of data over its lifecycle. Data integrity establishes recoverability, searchability, traceability to origin, and connectivity. Protecting the validity and accuracy of data also increases stability and performance while improving reusability and maintainability.

Many cryptographic protocols now, go beyond these conventional goals to incorporate other aspects like key-agreement or establishment, non-repudiation methods, secure multi-party computation, blind signatures, secure digital timestamping, etc.

- **Key agreement or establishment** refers to the process of establishing a shared secret key between two or more communicating entities. These entities can agree on a key in such a way that both influence the outcome. If properly done, this method of using keys prevents undesired third parties from forcing a key choice on the communicating entities.

- **Non-repudiation** is a method to ensure that a communicating entity cannot deny the authenticity of their communication, i.e., a sender of a message cannot deny having sent the message and a recipient cannot deny having received the message. Non-repudiation services are commissioned to collect and validate evidence regarding a claimed event or action to resolve disputes concerning the occurrence or non-occurrence of the event.

2.2 Security in Protocol Design

It is a difficult task to design cryptographic security protocols accurately. The security of cryptographic protocols is an extremely sensitive topic. Various security protocols believed to be strong have been identified later to contain flaws and weaknesses. To attain the security property that cryptographic protocols must fulfill, concepts like nonce, session keys or timestamps are employed.

- **Session keys:** Session keys are single-use symmetric keys that are used for all the message exchange in a communication. Session keys are mostly chosen at random to prevent the adversary from predicting them. Communicating entities can mutually agree on a session key by exchanging several messages. The entities can then use this session key to exchange encrypted messages.

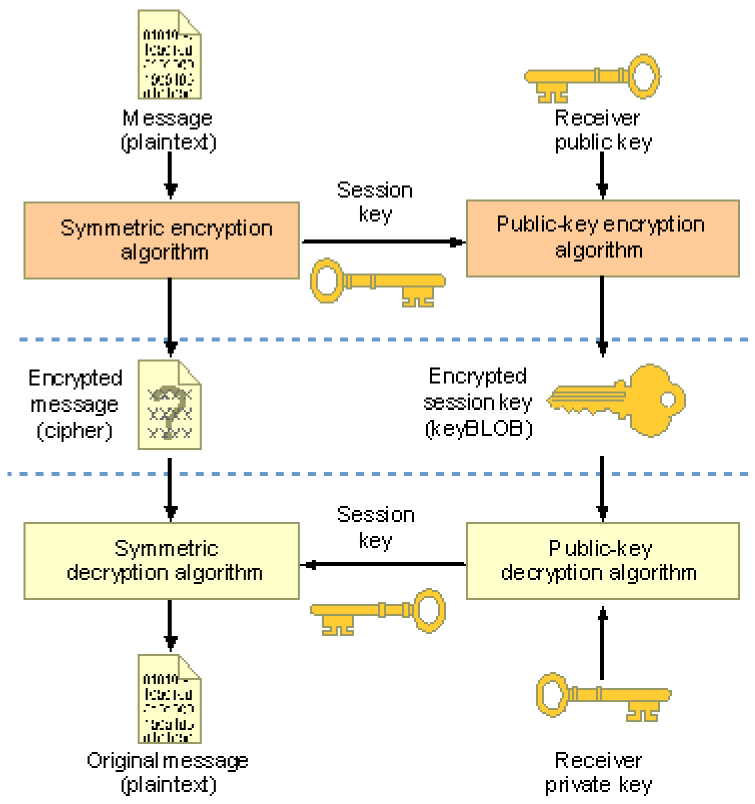

Message (plaintext)
Receiver public key
Session key
Symmetric encryption algorithm
Public-key encryption algorithm
Encrypted message (cipher)
Encrypted session key (keyBLOB)
Session key
Symmetric decryption algorithm
Public-key decryption algorithm
Original message (plaintext)
Receiver private key

Figure 1 : Exchanging session keys *[6]*

- **Nonce:** Nonces are one of the most common approaches used by protocol designers. A nonce, usually a random or pseudo- random number, is an arbitrary number used only once to sign a cryptographic message belonging to one protocol session. It is issued in an authentication protocol to ensure that old communications cannot be reused in a replay attack. A nonce is known only by its generator and it is use to verify the authentication and freshness of a message. When we use a nonce to prevent a replay attack, we need a way of knowing that we have handled a message with the nonce value already.

- **Timestamps:** Timestamps are used to ensure freshness of messages exchanged during the protocol run. A timestamp is a sequence of characters which expose the date or time when a certain message was sent. A trusted timestamp is usually generated by a Trusted Third Party (TTP) acting as a Time Stamp Authority (TSA). It is used to express the validity of a data before a certain time. Its creation is based on digital signatures and hash function. First, a hash is calculated from the data to be sent and this hash is sent to the TTP. The TSA concatenates a timestamp to the hash and builds a new hash out of this concatenation. The latest hash is digitally signed with the private key of the TTP and the signed hash and the timestamp are sent back to the timestamp requester.

2.3 Attacks on Security Protocols

An attack is an action that exploits the flaws and vulnerabilities in the design of a security protocol to gain unauthorized access to information. It occurs when an intruder or adversary compels a communicating entity to compromise the security of the exchanged messages in a communicating system. All the aspects of security, namely confidentiality, integrity and authentication are compromised in an attack.

An intruder has full control over the insecure network and can eavesdrop, insert, delete, modify, intercept and synthesize messages.

An attack may be *active* or *passive.*

- In an **active attack**, the intruder attempts to change system resources or influence their operation.
- A **passive attack** tries to learn or make use of information from the system but does not affect system resources.

An attack could be an *inside attack* or an *outside attack*.

- **Inside attacks** are initiated by entities authorized to access system resources but uses them in a way not approved by the authorizers.
- **Outside attacks** are initiated outside the security perimeter by an illegitimate or unauthorized user.

2.3.1 Replay attacks

Data freshness is one of the security requirements of any cryptographic protocol - the content of the messages exchanged between the principals should belong to the current session or protocol run. In other words, the data generated for a certain session should not be valid or accepted by the principals in any other session of the protocol, so that an attacker who might gain access to a message or parts of a message could not use them in a replay attack.

A replay attack takes place when an attacker or adversary copies a stream of messages between two communicating entities and replays the stream to one or more of these entities. Unless mitigated, the distributed system under the attack process the stream of messages as legitimate messages, resulting in a range of adverse and undesirable consequences.

For example, let us assume an entity X wants to prove its identity to entity Y. Entity Y then requests the password of X as a proof of identity, which X provides using a transformation like a hash function. Meanwhile, intruder Z is eavesdropping on the exchange and keeps the password, or the hash of the password. After the interchange is over, Z, posing as X, connects to Y. When asked for a proof of identity, Z sends the password or the hash of the password of X read from the last session which Y accepts, thus granting entity Z access.

2.3.2 Parallel Session Attacks

A **parallel session attack** requires the existence of parallel execution of multiple protocol runs. Messages from one session are used to synthesize messages in another session by the attacker. Examples of parallel session attacks on security protocols include [7] [8] [9] [10]. There are many forms of parallel session attacks:

- **Man-in-the-middle-attack:** The attacker can intercept the messages exchanged between the honest participants and can inject new ones. It is a form of eavesdropping when the attacker makes random connections between the victims, relaying messages between them but making them believe they are talking with each other over a secure connection. A man-in-the-middle attack can succeed only if the attacker can impersonate the participants. The attack utilizes a lack of mutual authentication between the participants.

- **Multiplicity attack:** It is a parallel session attack where the principals disagree on the number of runs they have successfully established with each other.

- **A type flaw attack:** A type flaw attack occurs when a recipient accepts a message as valid, but interprets the bit sequence differently from the creator of the message. It involves the replacement of a component included in one message with another message, of a different type by an intruder.

- **Oracle attack:** In an oracle attack the intruder starts a new run of the protocol and uses one of the participants as an oracle for appropriate answers to challenges in the first protocol run.

Chapter 3: Security protocols for mobile communication

Communication security is often described in terms of security services such as authentication, confidentiality, integrity, and nonrepudiation of transmitted data which are in turn implemented by various mechanisms that are usually cryptographic in nature. However, mobile communication systems have various additional security requirements apart from the traditional requirements due to the characteristics of the mobile environment. These include the communications path, location privacy and computational constraints. Some part of the communication path such as the radio link is particularly vulnerable to attack. Also, it is necessary to limit the computational complexity of security protocols as mobile devices are mostly computationally limited.

Mobile communication networks have many limitations in comparison with fixed networks [11] :

- Open mobile access medium: There is no physical barrier between an attacker and the network as the communication is through a mobile channel.
- System complexity: Mobile systems are more complex due to the need to support mobility and making use of the channel effectively. By adding more complexity to systems, potentially new security vulnerabilities can be introduced.
- Limited power: Mobile Systems consume a lot of power and therefore have a limited time battery life.
- Limited bandwidth: Although wireless bandwidth is increasing continuously, because of channel contention everyone must share the medium.
- Limited processing power: The processors installed on the wireless devices are increasing in power, but still they are not powerful enough to carry out intensive processing.
- Relatively unreliable network connection: The wireless medium is an unreliable medium with a high rate of errors compared to a wired network.

A generic set of security requirements for protocols in mobile communication systems is as follows [11] :

- Mutual Authentication of user and network
- Exchange of certified public keys
- Session key agreement

- Joint control of session key
- Mutual implicit key authentication
- Mutual key confirmation
- Mutual assurance of key freshness
- Confidentiality
- Initialization of payment mechanism
- Non-repudiation of origin

The infrastructure is open to various types of attacks as the mobile communication system is an open system and has a huge architecture. An attacker may try to break the encryption of the mobile network or try to eavesdrop on Wi-Fi communications to derive information. Common attacks are denial of service, unauthorized access, channel jamming, eavesdropping, message replay, message forgery, man in the middle attack and session hijacking.

For this project, nine recent security protocols for wireless/mobile communication have been considered. There are three protocols published in the year 2017, four of them were published in 2016, and the other two in 2015. The following subchapters are descriptions of how these nine protocols work.

3.1 Classiquantum Resistance Secure Simple Pairing (CRSSP) - 2017

Classiquantum Resistance Secure Simple Pairing (CRSSP) [12] is a quantum resistance Bluetooth pairing protocol for quantum computers that uses Lattice-based Ring Learning with Errors (R-LWE) to enhance the key strength by wrapping a link key with two R-LWE secret keys. The CRSSP protocol overcomes the limitations of the existing Secure Simple Pairing (SSP) protocol introduced in Bluetooth v2.1 by improving the security of pairing and authentication process. A major limitation of SSP protocol is that the inverse of Elliptic Curve Discrete Logarithm can be estimated in polynomial time using an algorithm for quantum computers.

CRSSP simulates numeric comparison mode with five phases of pairing:

- Phase 1 - Input-Output capability exchange,
- Phase 2 - R-LWE Public Key Exchange,
- Phase 3 - Signature Generation and Verification,
- Phase 4 - Second phase of R-LWE Public Key Exchange
- Phase 5 - Link/Session Key generation

The working of CRSSP protocol is as follows:

1. Devices A and B exchange their unencrypted IO capabilities and Bluetooth Device Addresses (BDAddr).
2. A and B compute the R-LWE Diffie–Hellman (R-LWE DH) public key and swap the public keys.
3. Both calculate R-LWE shared secret key from its own private key and others' public key.
   - SecretKey1 = *f1 (RLWEPrivateKey1 , RLWEPublicKeyAlice1, RLWEPublicKeyBob1 , RLWEMaskingBits1)*
   - SecretKey2 = *f1 (RLWEPrivateKey2 , RLWEPublicKeyAlice2 , RLWEPublicKeyBob2 , RLWEMaskingBits2)*
4. Each device A and B randomly selects a pseudo-random number of 128-bit and send the same to each other.
5. Based on the secret key, nonce (random number) and public keys, A and B computes their digital signature.
   - Signature = *g (RLWESecretKey1 , NonceAlice, NonceBob, RLWEPublicKeyAlice, RLWEPublicKeyBob)*
6. Each device passes the digital signature to its connecting device in the piconet and they substantiate digital signatures of each other. The protocol proceeds further, only when signatures are verified.
7. If substantiation fails, pairing begins again from step 1.

8. Once verified, both devices perform a new round of public key exchange and computes the R-LWE shared secret key again with new public keys and masking bits, following the same procedure as mentioned in steps 2 and 3.
9. Subsequently, both the connecting devices A and B calculate link/session key from secret keys obtained in the last step, the publicly exchanged data, their device addresses and IO capabilities.
   - SessionKey = *h (RLWESecretKey1 , RLWESecretKey2 ,RLWEPublicKeyAlice2 , RLWEPublicKeyBob2 , bdAddressAlice, bdAddressBob, ioCapAlice, ioCapBob)*
10. This link/session key is further used to encrypt messages or files that are to be sent.

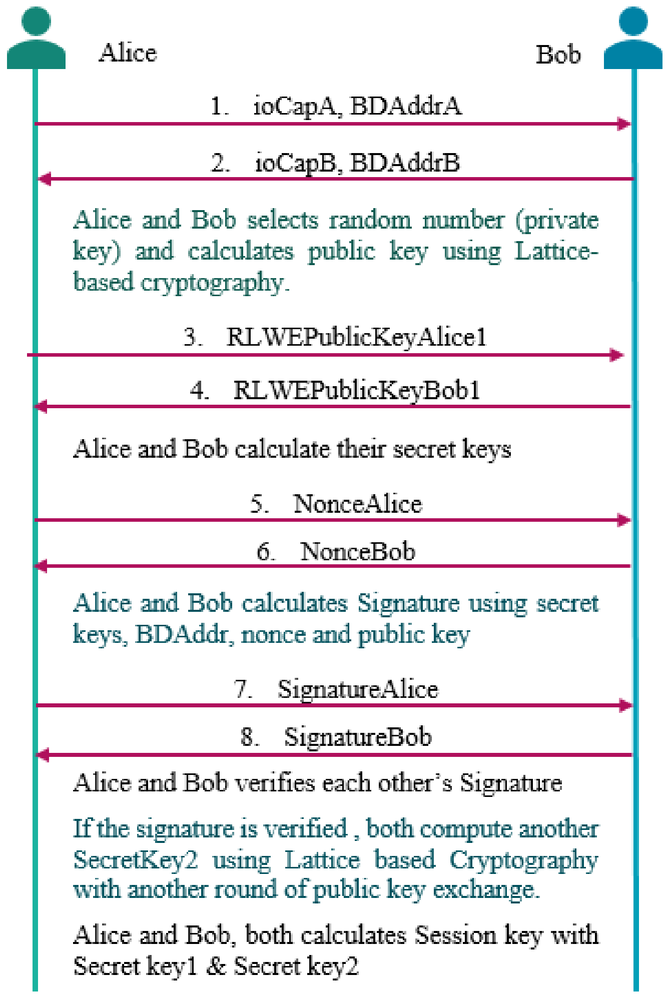

Alice
Bob
1. ioCapA, BDAddrA
2. ioCapB, BDAddrB
Alice and Bob selects random number (private key) and calculates public key using Lattice-based cryptography.
3. RLWEPublicKeyAlice1
4. RLWEPublicKeyBob1
Alice and Bob calculate their secret keys
5. NonceAlice
6. NonceBob
Alice and Bob calculates Signature using secret keys, BDAddr, nonce and public key
7. SignatureAlice
8. SignatureBob
Alice and Bob verifies each other's Signature
If the signature is verified , both compute another SecretKey2 using Lattice based Cryptography with another round of public key exchange.
Alice and Bob, both calculates Session key with Secret key1 & Secret key2

Figure 2: Working of CRSSP protocol

3.2 HIP-based Secure Software Defined Mobile Networks - 2017

A Software Defined Mobile Network (SDMN) architecture implements the use of software in the design of mobile networks and improves the performance, pliability and adaptability of present day telecommunication systems. SDMN is a flow-centric mobile network design based on the decoupling of data and control in the mobile network user plane and combines the core principles of Software Define Networking (SDN) and Network Function Virtualization (NFV) providing network programmability, centralized controlling and flexibility.

However, there are various challenges in the security of SDMN communication channels, i.e, control and data channels. Using IPsec tunneling mechanisms and security gateways is a method to secure backhaul communication channels in mobile networks. The protocol [13] to secure SDMNs using Host Identity Protocol (HIP) utilizes IPSec tunneling mechanisms to secure mobile communication channels and HIP helps in mutually authenticating two end nodes by establishing a Security Association (SA) for IPSec tunnels. The following diagram [13] represents the secure communication architecture for SDMN using HIP.

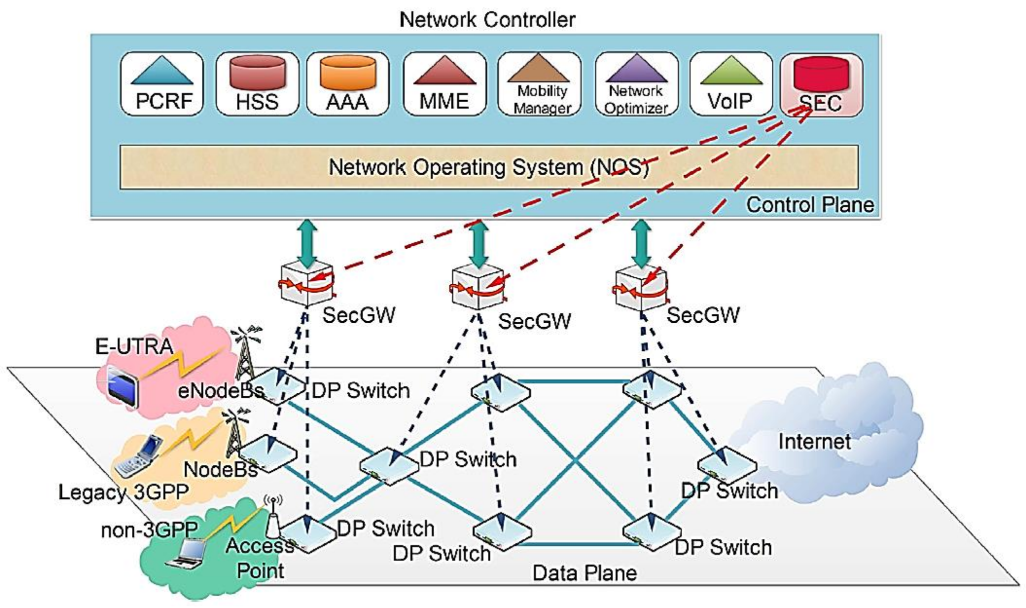


Figure 3: HIP-based secure SDMN communication channel architecture

The major components of the above-mentioned architecture are as follows:

a) Data Plane

- Consists of low-end switches, called data plane switches (DP Switches) and network links among them, which are connected to a centralized controller.
- Base stations, wireless access points and the Internet are connected to these DP switches (DPSs).
- User traffic is transported through the data plane.
- This communication channel is called the **data channel**.

b) Control Plane

- Consists of a logically centralized controller that controls the packet forwarding functions of the network through an open interface
- Also controls all the mobile backhaul functionalities such as routing, session initiations, session terminations and billing functions.
- The communication channel between the controller and DPS is called the **control channel**.

c) Distributed Security Gateways (SecGWs)

- Utilized to secure the controller from the outside network.
- Intermediate device between the controller and the data plane.
- Hides the network controller from the outside world and reduce the security related work load of the controller.
- Responsible for two functions:
  - ⇒ Establish IPsec tunnels with DPSs,
  - ⇒ Relay the messages between SecE and DPSs.
- Here, utilize distributed/multiple SecGWs to prevent a single point of failure.
- It is possible to integrate various security functions such as Intrusion Detection Systems (IDS), Deep Packet Inspection (DPI) and Firewalls within SecGWs to provide extra protection.

d) Security Entity (SecE)

- Added as a control entity to control the SecGWs and other security functions.
- New control entity which controls the SecGWs and other security functions.
- Authorizes DPSs based on Access Control Lists (ACLs).
- The network operator uploads a set of ACLs which contain the identities of legitimate DPSs.
- Also generates Traffic Encryption Keys (TEKs) for both control and data channels.
- Cooperates with other control entities (e.g. Traffic Optimizer Entity (TOE)) to manage the tunnel establishments in the control and data channels.

e) Local Security Agent (LSA)

- Installed in each DPS to handle security related functions in the switch
- Local SEC Agent (LSA) LSA is a security entity which is implemented in each DPS.
- Mainly responsible for HIP tunnel establishments with SecGWs and other DP switches.

3.2.1 HIP-based control channel

The HIP based protocol [13] uses IPsec Encapsulating Security Payload (ESP) Bounded-End-to-End-Tunnel (BEET) mode tunnels to secure the communication channels in SDMN. An HIP tunnel is established between SecGW and LSA as shown in the above given figure. Common control protocols are used for communication between the controller and DP Switches.

The HIP-based SDMN architecture supports the dynamic addition of new DPSs and the automatic control channel establishment. Hence, the two important tasks of the HIP-based protocol are:

1) Authenticate and register a new DPS.
2) Establish an IPsec (ESP BEET mode) tunnel between the DPS and SecGW.

The working of authentication and tunnel establishment procedure for an HIP-based SDMN control channel is as follows:

1. DPS initiates tunnel establishment procedure by sending **an I1 message** to SecGW. It contains the HITs (Host Identity Tags) of DPS and SecGW. HIT corresponds with the 128-bit hash of the HI.
2. To prevent DoS attacks, SecGW replies the I1 message with a pre-computed **R1 message** without allocating any resources.
   Components of the R1 message are:
   - *cryptographic puzzle,*
   - *D-H key parameters,*
   - *public key of SecGW,*
   - *ESP transforms,*
   - *HIP transforms,*
   - *echo request*
   - *signature*
3. DPS sends the **I2 message** to SecGW.
   Components of the I2 message are:
   - *HMAC (Hash Message Authentication Code),*
   - *solution of the cryptographic puzzle,*
   - *D-H key parameters,*
   - *public key of the DPS, SPIs,*
   - *ESP transforms,*
   - *HIP transforms*
   - *signature.*
4. SecGW sends the concerned DPS's credentials to SecE via **REQ message**.
   Components of the REQ message are:
   - *HIs of DPS and SecGW,*
   - *authentication request,*
   - *echo request.*

5. Upon the arrival of REQ, SecE checks the received HIs against the ACLs and the network optimizer. Then, it replies the REQ message with an **ACK message**.
   Components of the ACK message are:
   - *two HIs,*
   - *the acknowledgment*
   - *the echo reply*.

   Here, the HI of the DPS is checked with the ACLs by SecE. This step prevents unauthorized access to the network. Moreover, SecE keeps a record of the different requests with a time stamp. It helps to identify replay attacks. If a DPS sends premature requests again and again, those requests will be dropped.
6. The Traffic Optimizer checks the HIs of DPS and SecGW with the traffic optimization procedure.
   A positive acknowledgment is sent for a request from an authorized DPS and a negative acknowledgment is sent for other requests. In case of a negative acknowledgment, SecGW drops the connection request from the DPS.
7. Else, SecGW completes the tunnel establishment by sending the **R2 message**. It contains the HIs of DPS and SecGW, SPIs, HMAC and the signature.

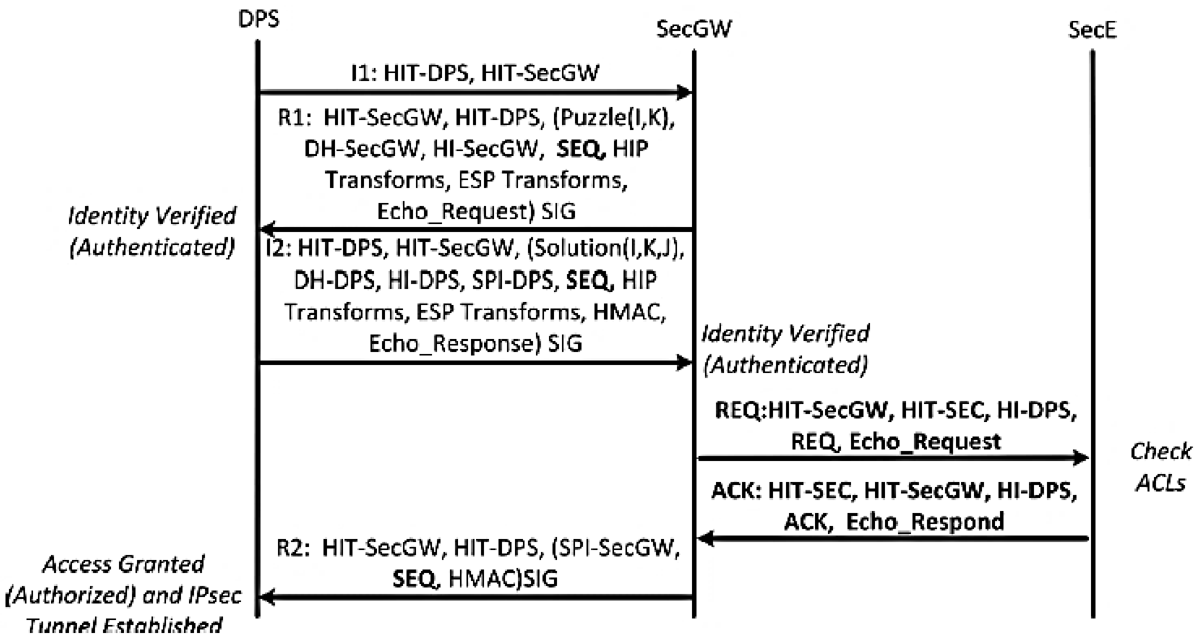

DPS
SecGW
SecE
I1: HIT-DPS, HIT-SecGW
R1: HIT-SecGW, HIT-DPS, (Puzzle(I,K), DH-SecGW, HI-SecGW, SEQ, HIP Transforms, ESP Transforms, Echo_Request) SIG
Identity Verified (Authenticated)
I2: HIT-DPS, HIT-SecGW, (Solution(I,K,J), DH-DPS, HI-DPS, SPI-DPS, SEQ, HIP Transforms, ESP Transforms, HMAC, Echo_Response) SIG
Identity Verified (Authenticated)
REQ:HIT-SecGW, HIT-SEC, HI-DPS, REQ, Echo_Request
Check ACLs
ACK: HIT-SEC, HIT-SecGW, HI-DPS, ACK, Echo_Respond
R2: HIT-SecGW, HIT-DPS, (SPI-SecGW, SEQ, HMAC)SIG
Access Granted (Authorized) and IPsec Tunnel Established

Figure 4: Tunnel establishment procedure for HIP-based control channel *[13]*

3.2.2 HIP-based data channel

In order to secure the data channel, HIP tunnels are established between LSAs. As the HIP-based protocol is invisible to Data Plane traffic, the traditional DP traffic is transported between DPSs without any modification.

Unlike the control channel, DPSs are mutually authenticated based on a Public Key Infrastructure mechanism before any data exchange. All the communication session establishment between DPSs is authorized based on ACLs and other control entities like traffic optimization entities. IPsec (ESP BEET mode) tunnels are established between switches to secure the data channel communication.

The tunnel establishment procedure for an HIP-based SDMN data channel is as follows:

1. DPS1 initiates tunnel establishment procedure by sending an **I1 message** to DPS2.
   It contains the HITs (Host Identity Tags) of DPS1 and DPS2.
2. DPS2 sends the **R1 message** to DPS2. Components of the R1 message is the same as that for control channel, with the only exception that instead of attributes of SecGW in control channel, the respective attributes of DPS2 are used here.
3. DPS1 responds with an **I2 message** to DPS2.
4. On reception of I2 message, DPS2 sends the **REQ message** to SecE via SecGW. It contains the HIs of DPSs and the echo request.
5. Upon the arrival of REQ, SecE checks the received HIs with ACLs and Traffic Optimizer.
6. Then, it replies the REQ message with an **ACK message**. The ACK message contains the HIs of the DPSs, the acknowledgment and the echo reply.
   A positive acknowledgment is sent for a request from authorized DPSs and a negative acknowledgment is sent for other requests. If it is a negative acknowledgment, DPS2 drops the connection request from DPS1. Otherwise, DPS2 completes the tunnel establishment by sending the R2 message.
7. As a final step of the tunnel establishment procedure, DPS sends the **R2 message** to DPS1.

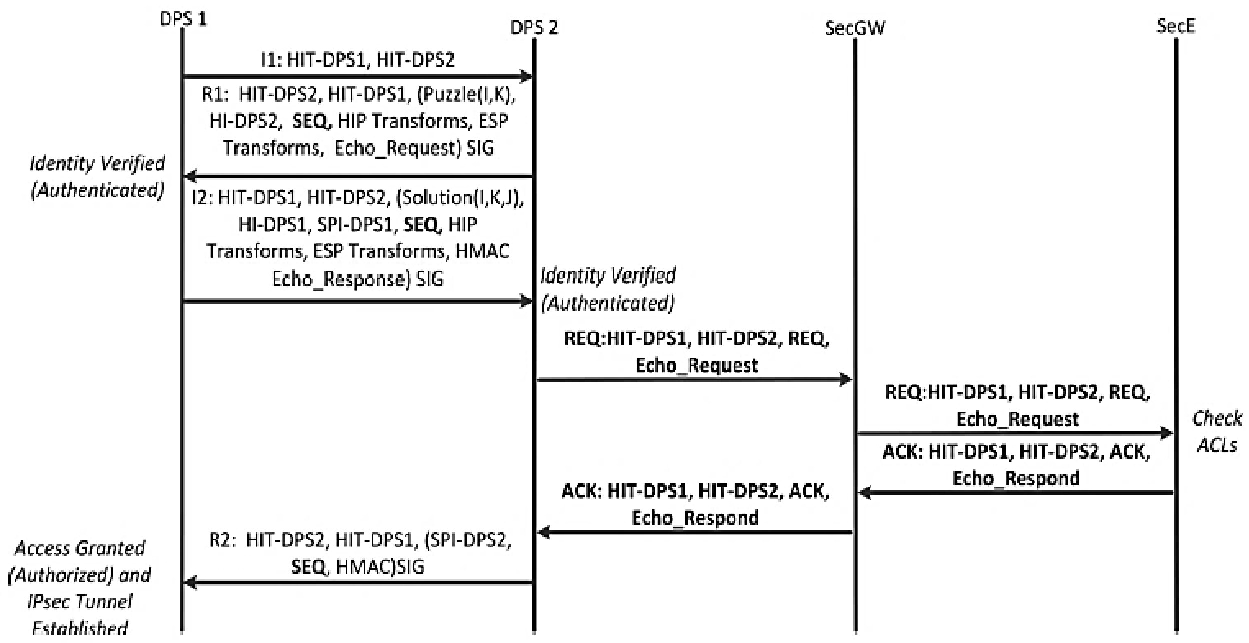

DPS 1
DPS 2
SecGW
SecE
I1: HIT-DPS1, HIT-DPS2
R1: HIT-DPS2, HIT-DPS1, (Puzzle(I,K), HI-DPS2, SEQ, HIP Transforms, ESP Transforms, Echo_Request) SIG
Identity Verified (Authenticated)
I2: HIT-DPS1, HIT-DPS2, (Solution(I,K,J), HI-DPS1, SPI-DPS1, SEQ, HIP Transforms, ESP Transforms, HMAC Echo_Response) SIG
Identity Verified (Authenticated)
REQ:HIT-DPS1, HIT-DPS2, REQ, Echo_Request
REQ:HIT-DPS1, HIT-DPS2, REQ, Echo_Request
Check ACLs
ACK: HIT-DPS1, HIT-DPS2, ACK, Echo_Respond
ACK: HIT-DPS1, HIT-DPS2, ACK, Echo_Respond
R2: HIT-DPS2, HIT-DPS1, (SPI-DPS2, SEQ, HMAC)SIG
Access Granted (Authorized) and IPsec Tunnel Established

Figure 5: Tunnel establishment procedure for HIP-based data channel

3.3 Secure Mobile Wallet – 2017

The rapid growth in technology has led to extensive usage of mobile devices in various fields, one among which is called mobile payment or mobile wallet. A mobile wallet is used to carry out payment transactions using a mobile device making it easy for mobile users, owing to the ubiquitous nature of mobile devices and Internet.

Being the most convenient payment method, mobile wallets are under scrutiny at present and have been found to possess various security challenges, mainly due to the open-medium of wireless communication and the limited resources of mobile devices.

The Secure Mobile Wallet protocol [14] is a method to secure a mobile wallet while preserving the privacy of a legitimate user by incorporating a certificateless digital signature scheme and pseudo-identity techniques. Also, to overcome the disadvantage of limited resources in mobile devices, this protocol employs the unlimited resources of cloud computing by outsourcing computation securely.

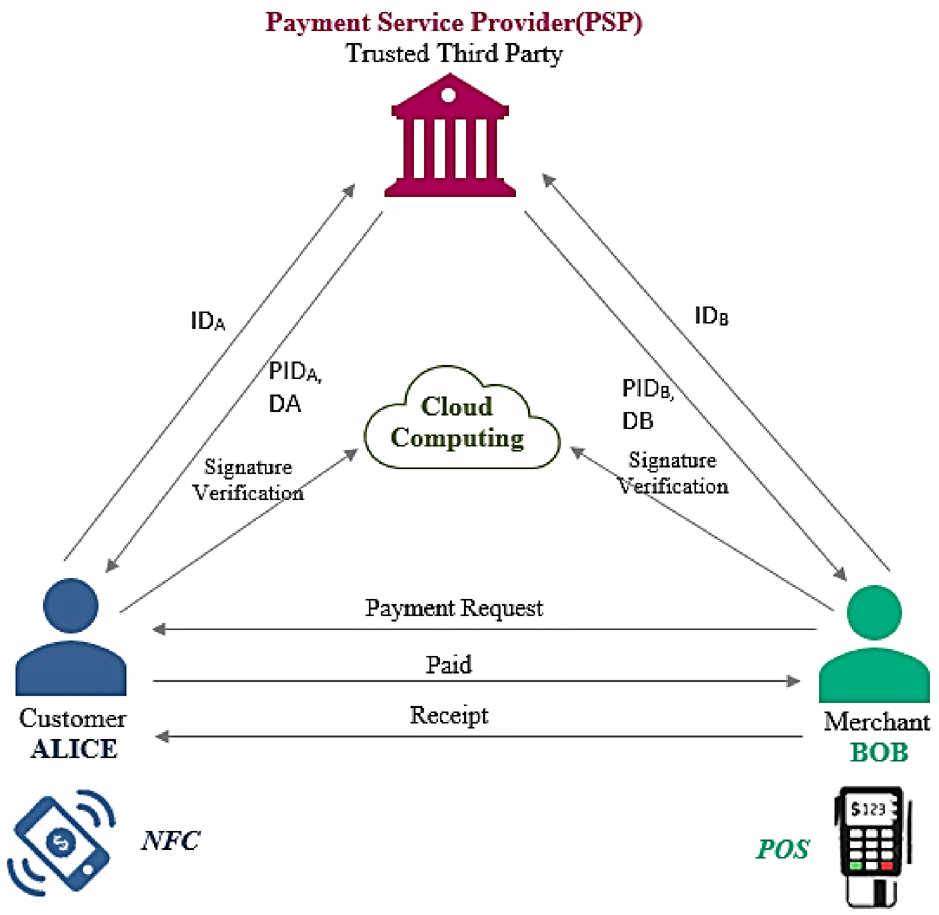


Figure 6: Working of mobile wallet

The major components in the model of a mobile wallet are [14]:

- **Customer** (Alice): The consumer who wants to purchase goods or services provided by merchant.
- **Merchant** (Bob): The merchant Bob who wants to sell goods or services to consumers.
- **Payment Service Provider** (PSP): A trusted third party called payment service provider PSP is responsible for the security and privacy of the payment information.

- **Untrusted Cloud Server Verification Provider** (CSVP): This entity reduces the computation overhead at user side by outsourcing the computation of verification to CSVP.
- **Wallet App**: An application that allows the user to perform payment transaction.
- **NFC-POS technology**: Near-field communication (NFC) is the basis for mobile wallet and refers to a set of communication protocols that enable two electronic devices, one among which is usually a portable/mobile device, to establish. The communication protocol between mobile device and merchant is assumed as NFC-POS, POS referring to Point-of-Sale.

The working of a secure and privacy-preserving mobile wallet [14] is:

1. Alice sends its real identity ID_A to the trusted third party, Payment Service Provider (PSP). The PSP stores this real ID_A in secure cloud storage.
2. PSP calculates the pseudo-identity of Alice PID_A and partial private key DA and sends these to Alice.
3. Bob sends its real identity ID_B to the trusted third party, Payment Service Provider (PSP). The PSP stores this real ID_B in secure cloud storage.
4. PSP calculates the pseudo-identity of Bob PID_B and partial private key DB and sends these to Bob.
5. Bob initiates payment request to Alice which contains the transaction identity TID, the amount to be paid, AmountT, and the pseudo-identity of B, PID_B,

   i.e., payment = $(TID \parallel AmountT \parallel PID_B)$
6. On receiving payment request from Bob, Alice uses its own private key and partial private key DA to generate a signature,

   $SignA = DA + H(payment \parallel PID_A \parallel PublicKey_A)\ PrivateKey_A$

   where, payment = $(TID \parallel AmountT \parallel PID_B)$
7. Alice then sends to Bob its signature SignA, the payment, and its public key.
8. On receiving Alice's SignA and public key, Bob verifies the signature using few calculations.

9. Once the signature is verified, Bob sends to Alice a receipt containing the transaction TID, the amount Bob received AmountR, and its signature SignB, calculated as,

$$\text{SignB} = \text{DB} + \text{H}(\text{receipt} \parallel \text{PID}_B \parallel \text{PublicKey}_A)\ \text{PrivateKey}_B$$

where, receipt = (TID || AmountR)

3.4 Secure patient-health monitoring Wireless Body Area Networks - 2016

Wireless Body Are Networks (WBANs) are an emerging technology extensively used nowadays in the medical field for long term patient-health monitoring. WBAN is essentially a wireless network of wearable or implanted sensors and the data sink which can be a smartphone or an implantable medical device that stores data. Sensors report all data to the data sink through wireless communication channels. There is a need to secure these channels to assure the privacy of the data being transferred.

Secure patient-health monitoring WBAN protocol [15] is a secure data communication scheme between sensors and data consumers (doctors or nurses) by employing Ciphertext-Policy Attribute Based Encryption (CP-ABE), two-phase commitment and session key authentication in wireless body area networks. Also, signatures are used to store the data in ciphertext format (encrypted format) in the data sink to ensure data security. The protocol specifies the sensors creating an access structure that allows only authorized doctors or consumers to access the data containing patient's private health information.

The major components in the architecture of a WBAN are [15]:

1. Key Generation Center (KGC)
   KGC generates the public parameters that are to be installed into the sensor before being implanted into human body. KGC carries out the task of system initialization by generating public parameters and allocating unique secret keys for all the data consumers. Each sensor creates an access structure or access tree using these public parameters that encrypts data according to the data tree. Here, the access of data by users at different privilege levels are performed using encryption.

2. Sensors

   The sensors employed in a WBAN can be implantable or wearable. The sensor collects all data regarding the patient's health and sends these in ciphertext form to the data sink.

3. Data Sink

   Data Sink can be a smartphone or an implantable medical device with access to Internet. It receives information from the sensors. This data is in ciphertext form ensuring the security of data. The compromise of a data sink would not mean the compromise of data stored in it. Doctors and other consumers access data from the data sink.

4. Data consumers

   Data Consumers can be doctors and nurses or other experts. The data consumers must have the attributes that satisfy the access tree to be able to decrypt a message obtained from the data sink.

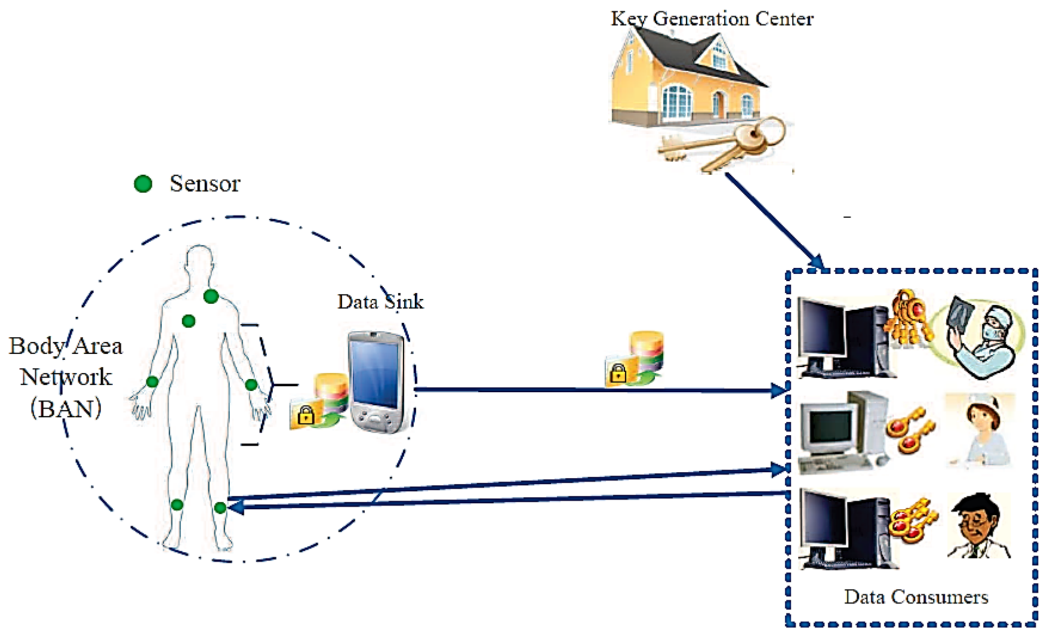


Figure 7: WBAN architecture

The working of secure patient-health monitoring WBAN protocol is as follows:

Phase 1: Communication Establishment phase

1. The sensor selects a random access token, Ksd and appends it with a timestamp TS to form K_{Tdate}, i.e, K_{Tdate} = Ksd || TS. Then it encrypts K_{Tdate} with the consumer's (here, doctor's) public key. Sensor updates K_{Tdate} after certain time interval.
   The sensor sends its identity IDs, encrypted K_{Tdate} and hash of K_{Tdate} to the data sink.

   Sensor -> Data Sink : IDs, E(K_{Tdate}), H(K_{Tdate})
2. The data sink stores the ciphertext form of K_{Tdate} and forwards the sensor's identity and the encrypted K_{Tdate} to the doctor.

   Data Sink -> Doctor : IDs, E(K_{Tdate})
3. The doctor decrypts E(K_{Tdate}) and obtains the value of K_{Tdate} and Ksd. It forms hash of this K_{Tdate} and sends it to the sensor to convince the sensor that he has the required privilege, along with its own identity, IDd.

   Doctor -> Sensor : IDd, H(K_{Tdate})
4. The sensor then compares this hash value to the one it already generated. If both are verified to be the same, then the doctor is an authorized user. The sensor then generated a new access token, Ksd1, and appends it with timestamp to obtain K_{Tdate1}. It sends this to the data sink to overwrite the previous access token, along with IDs, and hash of $K_{Tdate1.}$

   Sensor -> Data Sink : IDs, E(K_{Tdate1}), H(K_{Tdate1})
5. The sensor sends its identity, and the encrypted K_{Tdate1} to the doctor as a challenge.

   Sensor -> Doctor : IDs, E(K_{Tdate1})
6. The doctor decrypts *E(K_{Tdate1})* and obtains Ksd1. It generates hash of this Ksd1 along with timestamp and sends it to the sensor as a response.

   Doctor -> Sensor : IDd, H(K_{Tdate1})
7. The sensor verifies if this true. If true, the sensor and doctor goes to the next phase.

Phase 2: Communication phase

8. Doctor sends instructions to sensor using shared secret key, Ksd1.

   Doctor -> Sensor : IDd, IDs, E_{Ksd1}(INSTRUC), H(Ksd1, INSTRUC, IDs)

9. Sensor decrypts message and obtains INSTRUC. It forms hash of Ksd1, INSTRUC and IDs and verifies if it is the same as the one obtained. If true, the message integrity has been maintained. All future messages are sent using Ksd1.

3.5 P2P service security protocol in M2M environment - 2016

Machine-to-machine (M2M) communication is used where tasks are repetitive and long-term, and human involvement is not viable. M2M communication systems send and receive information over a wireless medium and thus are prone to attacks from intruders. Peer-to-peer (P2P) service security protocol [16] performs mutual authentication of machines and ensures security in M2M communication. This communication protocol [16] is designed by applying hash locks, random numbers, and session keys.

The security requirements of M2M communication are [16]:

- Each entity has a unique ID that is used to identify itself during M2M communication.
- Confidentiality of information should be maintained so that intruders cannot access data being transmitted.
- It is necessary to maintain anonymity in M2M communication as it can avoid the risk of intrusion on privacy.
- Digital signatures are used for non-repudiation in M2M communication.
- Privacy is required to prevent ID, video data and location information from being exposed to intruders.
- If any information is recursively used in M2M communication, locations of devices must not be exposed, owing to the feature of unlinkability.
- In case of crimes and incidents arising, relevant information should be traceable, whilst trace information should be thoroughly secured.

The working of P2P service security protocol in M2M communication is as follows:

1. After receiving the query from Bob, Alice generates a hash function of Bob's identity BobID, timestamp TSa, and BobID, Alice's secret Secret(Alice), session key of Alice, Ka, all encrypted using database server's public key.

   Alice -> Bob : H(BobID), TSa, $E_{DBPublicKey}$(BobID, Secret(Alice), Ka)
2. On receiving Alice's message, Bob calculates hash of BobID, timestamp TSb, and AliceID, Bob's secret Secret(Bob), Bob's session key, Kb, encrypted using database server's public key. Bob forwards this along with the message obtained from Alice to the Database Server.

   Bob -> DBServer : H(BobID), TSb, $E_{DBPublicKey}$(AliceID, Secret(Bob), Kb), $E_{DBPublicKey}$(BobID, Sercret(Alice), Ka)
3. DBServer receives two messages from Bob, one containing BobID and the other containing Alice ID. DBServer performs mutual authentication using these two messages. It then concatenates BobID with its own ID, idDB, to form hash of BobID and idDB.

   DBServer -> Bob : H(idDB, BobID), Ka (+) Kb
4. Bob checks and authenticates value received from DB and sends to Alice BobID encrypted using Ka and hash of idDB for authentication.

   Bob -> Alice : E_{Ka}(BobID), H(idDB)
5. Upon completion of checking the value, Alice encrypts her own ID with hash operation, transmits it to Bob, and finishes the authentication session in Alice.

   Alice -> Bob : H(Ka, AliceID), TSa

3.6 Secure communication in vehicular networks - 2016

Vehicular communication is an emerging technology and is being used in many vehicles nowadays with onboard sensors, communication devices, resource rich hardware, computing power, storage and internet connectivity. The sensors send all information to on-board units (OBU). Each on-board device has wireless communication to transfer these information to road-side units (RSU). The authenticity of the information being shared and secure communication between vehicles and road-side units are addressed in this secure communication protocol [17]. The secure and lightweight communication protocol for vehicular communication [17] is a key pre-distribution and establishment protocol where a shared key is deduced at the end of key transfer sessions.

This lightweight protocol assigns a hierarchy model which consists of

- Trusted authority (TA) as Root Node (Level 0)
- Road-side unit coordinator (RSU-C) at Level 1
- Road-side units (RSU) at Level 2 and
- Vehicles as Leaf nodes (Level 4)

The working of lightweight communication protocol for vehicular networks is as follows:

1. OBU sends to RSU its unique ID key, ELP (electronic license plate) encrypted using RSU's public key.

   OBU -> RSU : $E_{RSUPublicKey}(ELP)$
2. RSU decrypts the received message and obtains ELP. RSU then encrypts using ELP, coefficients of polynomial and an ID selected by RSU from a pool of IDs for an OBU.

   RSU -> OBU : E_{ELP}*(Coeffs of polynomial, ID)*
3. The shared key is then computed as a function of the coefficients of polynomial and ID.

3.7 TTP Based High-Efficient Multi-Key Exchange Protocol (THMEP) - 2016

Trusted-third-party-based High-efficient Multi-Key Exchange Protocol (THMEP) [18] is a reliable and scalable key exchange method for data communication, where a trusted third party (TTP) is used to authenticate entities and their messages. THMEP makes use of elliptic curve cryptography and a current time encryption key to exchange session keys between entities. It also performs mutual authentication between a user and TTP. THMEP resists known-key, impersonation, replay, eavesdropping, and forgery attacks as it generates 40 session keys in a key exchange process.

THMEP consists of two phases – initialization phase and authentication and key exchange phase. In the initialization phase, the TTP generates public parameters and hash functions, user A selects its own password pwA and user B selects its password pwB from the password space D. TTP then derives the password key *Kpwa* from *pwA* for user A and password key *Kpwb* from *pwB* for B.

The working of authentication and key exchange phase of THMEP [18] is as follows:

Round 1

1. TTP (Trusted server S) selects random numbers Nsa and Nsb. S computes SA = {Nsa}KaPub, s'A = F(H(Nsa),Kpwa) and SB = {Nsb}KbPub, s'B = F(H(Nsb),Kpwb).
2. S then sends identity of S IDs, timestamp TSs, s'A and SA to A.

   S -> A : {IDs, TSs , s'A, SA}
3. S sends IDs, timestamp TSs, s'B and SB computed earlier to B.

   S -> B : {IDs, TSs , s'B, SB}

Round 2

4. A selects random number Na and uses password Kpwa to compute XA = {Na}KsPub and x'A = F(H(Na),Kpwa). A sends A's identity IDa, x'A and XA.

   A -> S : {IDa; x'A; XA}
5. B selects random number Nb and uses password Kpwb to compute XB = {Nb}KsPub and x'B = F(H(Nb),Kpwb). B sends B's identity IDb, x'B and XB.

   B -> S : {IDb; x'B; XB}

Round 3

6. S computes nA which is a hash of IDa, IDs, IDb, Na, Nsa, XB, SB and Kpwa. S sends its IDs, B's identity IDb, XB, SB and nA to A.

   S -> A : {IDs, IDb, XB, SB, nA}

   nA = H(IDa; IDs ; IDb; Na; Nsa; XB; SB; Kpwa)
7. S computes nB which is a hash of IDb, IDs, IDa, Nb, Nsb, XA, SA and Kpwb. S sends its IDs, A's identity IDa, XA, SA and nB to B.

   S -> B : {IDs, IDa, XA, SA, nB}

   nB = H(IDb; IDs ; IDa; Nb; Nsb; XA; SA; Kpwb)
8. A and B generates session key at their own sides with these parameters.

3.8 MAKA scheme with user anonymity in GLOMONET – 2015

The MAKA (Mutual Authentication and Key Agreement) scheme [19] is a privacy-preserving or user anonymous authentication scheme for roaming services in Global Mobility Networks (GLOMONET). The MAKA scheme focuses on protecting GLOMONETs against eavesdropping, forgery, known session key attacks, etc. Along with achieving mutual authentication with user anonymity, MAKA scheme also reduces the computation and communication cost in mobility networks. There are three phases in MAKA scheme. Phase 1 is the registration phase, phase 2 is the mutual authentication and key agreement phase, and phase 3 is the password renewal phase.

In the registration phase, the mobile user MS receives a smartcard from home agent HA. In the MAKA phase, both the MS and foreign agent FA authenticate themselves through HA and establishes a session key between them. In the password renewal phase, the mobile user MS can renew its password.

The working of MAKA scheme [19] is as follows:

Phase 1: Registration Phase

1. MS sends to a particular HA the identity of MS, IDm. On receiving idM, HA generates random number Nh, and calculates Kuh = H(idM, Nh).

   MS -> HA : IDm

   HA : Kuh = H(idM,Nh)
2. HA sends to MS key Kuh, pseudo-identity HA generated for MS, PID, emergency key, Kem, and transaction sequence number, seq.

   HA -> MS : Kuh, PID, Kem, seq

Phase 2: MAKA Phase

3. MS then calculates AidM as H(idM, Kuh, Nm, seq), where Nm is the nonce generated by MS, Nx as F(H(idM, Kuh), Nm) and sends these to FA along with seq and IDh

   Ms -> B : AidM, Nx, seq, IDh

   Ms: AidM = H(idM, Kuh, Nm, seq)

   Nx = F(H(idM, Kuh), Nm)

4. FA forwards the message received from MS to HA, along with IDf, Ny calculated as F(H(Kfh), Nf) and V1 = H(AidM, Nx, seq, IDh, Ny, Kfh, Nf).

   FA -> HA : AidM, Nx, seq, IDf, V1, Ny

   B : Ny = F(H(Kfh), Nf)

   V1 = H(AidM, Nx, seq, IDh, Ny, Kfh, Nf)

5. HA sends to FA the following message:

   HA -> FA : N'x, N'y, V2, V3, Ts

   C : N'x = F(H(Kuh, idM, seq), Nf)

   N'y = F(H(Kf, Nf), Nm)

   V2 = F(H(N'y, Nf), Kfh)

   Ts = F(H(Kuh, idM, Nm), seq)

   V3 = F(H(N'x, Nm, Ts), Kuh)

6. FA sends to MS N'x = F(H(Kuh, idM, seq), Nf), V3 = F(H(N'x, Nm, Ts), Kuh) and TS.

   B -> Ms : N'x, V3, Ts

Chapter 4: Verification of Security Protocols

Although there is a requirement for security and privacy in active communication and information systems, the underlying security protocols used in these systems are assailable to a range of attacks such as replay attacks [20], freshness attacks [21], parallel session attacks [22] and repudiation. Hence, security protocols must be tested for functional correctness to ensure that these protocols fulfill their intended goals. Both formal and informal methods of security protocol verification have been proposed. Traditionally, cryptographic protocols have been designed and verified using informal and intuitive techniques [23]. However, informal techniques have proven to be less effective as it resulted in flaws remaining undetected. On the other hand, formal verification techniques provide an organized way of discovering protocol flaws. For example, the of Needham and Schroeder public-key authentication protocol [5] was considered secure for over a decade until verification of the protocol using formal logics [1] exposed vulnerability to replay attacks.

Since the late 1980's, formal methods have been used in the field of security protocols. Formal verification refers to the deployment of mathematical-based techniques to establish that the intended algorithm conforms to certain precisely expressed notion of correctness. Formal methods have been used to analyze protocols with varying levels of complexity. Any proof of correctness is always relative to the formal specification of the system, the required properties and to the assumptions of the formal system itself.

Formal verification is an essential part of the design process [24], as it:

- provides a systematic way to detect design flaws;
- identifies the exact cryptographic properties a protocol aims to satisfy;
- identifies the assumptions and the environment under which these properties hold;
- removes ambiguity in the specifications of the protocol.

Formal verification approaches include various techniques [25] such as state machine methods, algebraic term rewriting, theorem proving techniques and modal logics. State machine-based techniques [26] [27] [28] [29] and modal logics [30] [31] [32] [33] have been the most successful in detecting various protocol flaws and weaknesses.

4.1 State Machine-Based Technique

In the state machine technique, cryptographic protocols are modelled as a set of finite state machines. To validate the security of a protocol, either an exhaustive exploration of the state space (exploring all states and transitions in the model, by using smart and domain-specific abstraction techniques to consider whole groups of states) is carried out. State exploration methods model the protocol as a finite state system. An exhaustive search checks that all reachable states are safe.
Reachability analysis technique is useful in checking the correctness of a protocol with respect to its specification. However, it does not assure security from an active attacker. Limitations of state machine techniques include the requirement of drastic assumptions in order to keep the state space small, and its failure to identify subtle flaws such as replay attacks. Techniques based on state machines often fail to detect new flaws in protocols, but it can verify if a protocol contains a given flaw. State space searching is found to be inefficient and often, the space becomes too large and cannot be analyzed) and cannot find new errors.

4.2 Algebraic Term Rewriting Techniques

In algebraic rewriting technique, cryptographic protocols are modelled as algebraic systems, where algebra is used to express the state of the participants' knowledge of the protocol. This technique starts with the specification of the start state and is shown that an insecure state cannot be reached.

Algebraic term rewriting technique is a refinement over the traditional state machine technique as it combines the state machine approach with the ability to reason about advancement of knowledge, thereby being able to discover new weaknesses in the protocol. However, accounting to the complexity of this technique, and its failure to detect new messages introduced by the intruder, the algebraic term rewriting approach is not used extensively.

4.3 Theorem Proving Techniques

Theorem proving was used for proving security properties of protocols. Proofs in an interactive theorem prover typically require much human interaction, but allow one to prove any mathematically correct result [34]. One major advantage of theorem proving is that proofs are highly automated. One Isabelle/HOL (higher-order logic) theorem prover command can generate thousands of inferences. Small changes to protocols involve only small changes to proof scripts.

Although theorem proving tools eases the task of generating proofs, the theorem proving approach is not very efficient at finding attacks in flawed protocols. Also, the inexperience of the user could result in a failed protocol verification as a theorem prover requires the user to find the right path through the proof, which demands for intelligence, intuition and practice.

4.4 Modal Logics

The technique based on modal logics of belief and/or knowledge is called logic-based verification [35] of security protocols and involves the process of deductive reasoning. Logics have been designed to reason about protocols. The earliest, and best known, is the logic by Burrows, Abadi and Needham [1], generally referred to as BAN logic.

In BAN logic, the assumptions and goals of the security protocols are stated as beliefs of principals. A message contains a set of beliefs held by its sender. The logic provides various inference rules that define how a recipient's belief state may legitimately evolve upon receipt of a message. Informally, a message of a sender's beliefs can be received, but it may be encrypted. If a principal has the key, then the plaintext may be obtained. However, the principal has no assurance where the plaintext comes from, unless the message is signed. If a message is signed, then a principal may assume that the signing principal once said the plaintext. But he does not know when the message was said, unless it is fresh, in which case, the principal may assume the signer effectively, says the message at the same time it is received (the message is timely, and thus not a replay). Still, the message may not come from a principal trusted to have the authority to make such statements. If it does, then whatever the message asserts is considered accurate [36].

Chapter 5: Logic-Based Verification

In verification using modal logics, a set of inference rules and postulates are applied to the initial assumptions and message exchanges to deduce the desired protocol goals [37]. Several logics have been developed on the basis of BAN logic, such as Syverson and Van Oorschot [38], Zhang and Varadharajan [39], Coffey and Saidha [37]. A number of flaws have been detected by these logics in protocols previously considered secure.

Logic-based verification of security protocols proceeds in four steps [23]:

- **Formalization of the protocol messages**: Formally specify steps of a protocol in the language of the logic.
- **Specification of the initial assumptions**: Start with the initial protocol assumptions, build up logical statements after each protocol step using the logical axioms and inference rules.
- **Specification of the protocol goals**: Formally specify the desired protocol goals.
- **Application of the logical postulates**: Compare these logical statements with the desired protocol goals to see if the goals are achieved.

The main objective of logic analysis is to test whether the desired protocol goals can be deduced from the initial assumptions and message exchanges of the protocol. The first step is to transform the informal security protocol specification into the language of the logic by expressing each message as a logical formula. This technique is called idealization and is usually done manually and hence, is error-prone. This is a limitation of logic-based verification.

The second step includes specifying the initial protocol assumptions. While the formal description of the protocol attempts to show the purpose of the components of each message in order to avoid ambiguity, the assumptions reflects the beliefs of the principals involved.

The third step involves expressing the protocol goals in terms of the language of the logic. Then in the final step, logical postulates are applied and the message exchanges are examined in order to verify the four desired protocol goals. When few conclusions are found to be incompatible with the expected goals, this indicates that the protocol contains security flaws and weaknesses. If a weakness is discovered, the protocol should be redesigned and re-verified [23] [3].

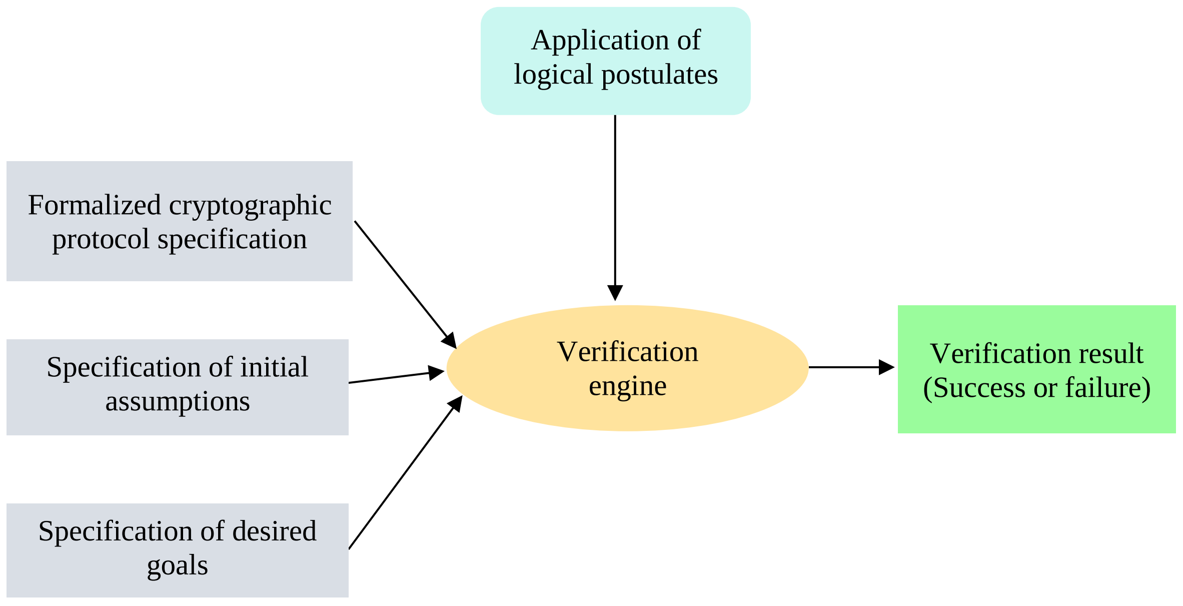

Application of logical postulates
Formalized cryptographic protocol specification
Specification of initial assumptions
Specification of desired goals
Verification engine
Verification result (Success or failure)

Figure 8: Logic-based verification

5.1 Automation of Security Protocol Verification

It is extremely difficult to manually perform logic-based verification as the application of postulates of a verification logic is tedious and error-prone. Hence, the automation of such a verification process can remove the potential errors in the manual process. Additionally, the automation process saves up on the time taken to perform verification.

Several possible sources of error are automatically removed, as an automated system will [40]:

- not make implicit assumptions
- not take shortcuts
- ensure thorough and unambiguous use of the postulates
- not make implicit assumptions about failed goals
- allow redundant assumptions to be identified easily

Automated verification tools, such as TAPS [41], AAPA2 [42] and PIL/SETHEO [43], are based on general theorem provers. Others, such as SPEAR II [44] and Hauser-Lee [45], are based on declarative programming languages [40].

- **TAPS - A First-Order Verifier for Cryptographic Protocols:** It is a verification method for cryptographic protocols, based on first-order invariants. For typical protocols, a suitable invariant can be generated mechanically from the program text, allowing safety properties to be proved by ordinary first-order reasoning. TAPS is an automated verifier that uses the method to prove safety properties comparable to those in published Isabelle verifications, but does so much faster with little or no guidance from the user.
- **AAPA 2 - Automatic Authentication Protocol Analyzer II:** It is a tool for analyzing cryptographic protocols that is based on an extension of the GNY logic. The AAPA2 automatically determines whether a protocol meets its specifications using the HOL proving engine.
- **PIL/SETHEO:** It is capable of automatically proving the security operations of protocols, formalized in the modal logic BAN and AUTOLOG. PIL/SETHEO uses the concept of meta-interpretation where each BAN or AUTOLOG formula is transformed into a term. All inference rules of the logic are translated into first-order implications.
- **SPEAR II - Security Protocol Engineering and Analysis Resource:** SPEAR II is a protocol engineering tool that focuses on enabling secure and efficient protocol designs and support for the production process of implementing security protocols.

Chapter 6: Cryptographic-Protocol Design & Verification Tool

The CDVT tool is developed at the Data Communication Security Laboratory of the Department of Electronic and Computer Engineering, University of Limerick, Ireland. The CDVT tool [3] is an automated system that can analyze the evolution of both knowledge and belief during a protocol execution and therefore is useful in addressing issues of both security and trust. The verification tool also has the capability of detecting protocol design weaknesses [46] that can be exploited by replay or parallel session attacks. It enables both attack detection analysis and conventional logic-based protocol verification from a single protocol specification.

The CDVT verification engine [3] is an automated logic-based verification tool that implements modal logics using the concept of Layered Proving Trees [2], and a parser to read in the formalized protocol specification from a text file. The CDVT tool will then apply postulates to the formalized specification and the outcome of this will allow the user to trace the performed proofs, thereby aiding the user in analyzing the reason for protocol failure, if a verification fails.

The CDVT tool displays the results of the protocol verification as a visualization of a layered proving tree. The proving tree may be expanded to examine details of the proof. This permits users to analyze how protocol goals are proven, or to establish why the protocol fails to satisfy its specified goals.

Layered Proving Trees are a specialized theorem prover for the needs of logic-based verification and it imitates the process used in manual application of logical postulates. If the application of logical postulates results in all the protocol goals being proven as valid, then the verification has succeeded and the protocol can be considered secure, provided that the initial assumptions and goals specified are correct. However, if the validity of some goals cannot be established, the verification has failed and it hints at problems or flaws in the protocol. The CDVT verification tool can be used for the following tasks [47]:

- Verification of cryptographic security protocols
- Detection of design weaknesses on security protocols
- Finding the reasons for failure of verified protocols, i.e., traceability of verification results
- Aiding in the re-design and correction of security protocols found to contain weaknesses

The CDVT tool comes in several versions: CDVT Attack Detection tool, CDVT Verification Logic tool, CDVT Verification Logic & Attack Detection and Combined CDVT tool. In the combined CDVT tool, users can choose what analysis to perform as it combines the three other modes in a single tool.

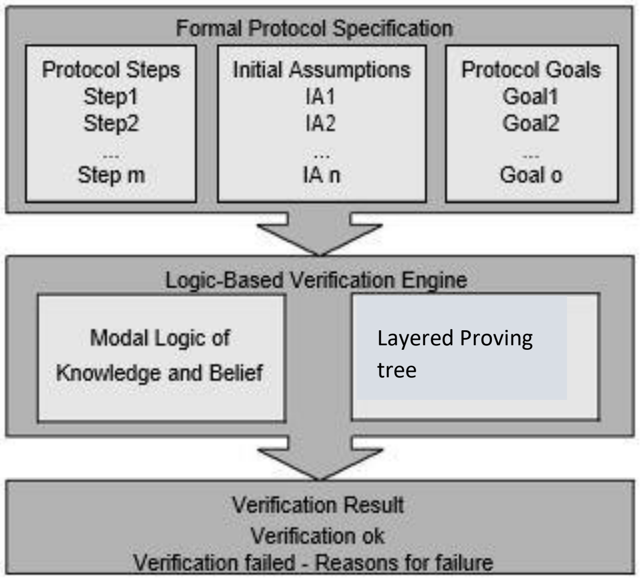


Figure 9: The CDVT verification tool *[23]*

The inference rules provided in the CDVT tool are the standard rules of natural deduction. The axioms of the logic express the fundamental properties of public-key cryptographic protocols, such as the ability of a principal to encrypt/decrypt based on knowledge of a cryptographic key. The axioms also reflect the underlying assumptions of the logic, which are as follows:

- The communication environment is reliable, but hostile. That is, message loss or modification can only occur as consequence of hostile intervention.

- The cryptosystem is ideal. That is, the encryption and decryption functions are completely non-invertible without knowledge of the appropriate cryptographic key and are invertible with knowledge of the appropriate cryptographic key. The cryptosystem is collision-free so that it is not possible to create the same ciphertext from two different pieces of plaintext.

- A public key used by the system is considered valid if it has not exceeded its validity period and only its rightful owner knows the corresponding secret key.

- If a piece of data is encrypted/decrypted, then the entity which performed the encryption/decryption must know that data (the data can be plaintext or ciphertext).

6.1 Input Language

The input language for CDVT tool is the formal specification of a security protocol formulated in an individual text file (.txt) and is processed by its logic of knowledge. The specification of a protocol consists of three parts – Assumptions, Steps and Goals, each of which follows well-defined syntax statements.

The general form of a specification is as follows,

Label : Statement ;

where *Label* is An, Sn, Gn for Assumptions, Steps, Goals respectively and 'n' is positive integer
(e.g. protocol assumptions are defined with label A1, A2, etc.,
protocol steps defined with S1, S2, etc.,
protocol goals defined with label G1, G2, etc.).

Statement takes one of the following structures:

- Principal Operator Time Data;
- Principal Operator Time Statement;
- Principal ComOperator Principal Time Data
- Principal know Time NOT (Statement);
- Principal believe Time NOT (Statement);

- (Statement);
- (Statement) AND (Statement);
- (Statement) OR (Statement);
- (Statement) IMPLY (Statement)

where *Principal* means either “an agent” or “trusted third party (TTP)” that follows the data format of a principal or trusted principal.

Operator can be “know”, “believe” or “possess”.

ComOperator can be “sendto” or “receivefrom”.

Time can be defined as “at [*index*]” where index is a number:

- -1 : indicates previous protocol runs;
- 0 : indicates the beginning of the current protocol run;
- 1 – n : indicates the time at step 1 – n.

The textual grammar used in CDVT tool is as follows [23]:

- Elements follow regular expressions as expressed in *Table 2.5.1*.
- Composite data components are constructed as in *Table 2.5.2*.
- Statement construction rules are defined in *Table 2.5.3*.

Textual Grammar	**Regular Expression**
Principal	[AB-EIJLMOQRSU-Z][A-Za-z_0-9_]*
Trusted Principal	TTP[A-Za-z0-9_]*
Sym. Key	K[a-z][a-zA-Z0-9_]*
Public Key	K[a-z][A-Za-z0-9_]*Pub
Private Key	K[a-z][A-Za-z0-9_]*Priv
Nonce	N[a-z][A-Za-z0-9_]*
Timestamp	TS[a-z][A-Za-z0-9_]*
Function	F[A-Za-z0-9_]*
Hash	H[A-Za-z0-9_]*
Binary Data	[a-z][A-Za-z0-9_]*

Table 1: Atomic units of textual grammar

Composite Data	**Textual Representation**
Concatenation	Data,Data
Group Element	(Data)
Symmetric Encryption	{Data}Data
Public Key Encryption	{Data}KPub
Private Key Encryption	{Data}KPriv
Function of Data	F(Data)
Hash of Data	H(Data)
Key Material of Data	KMaterial(Data)

Table 2: Composite Data Construction

Principal Operator at[i] Data
Principal Trans_Operator Principal at[i] Data
Principal know at[i] Statement
Principal believe at[i] Statement
Principal know at[i] NOT (Statement)
Principal believe at[i] NOT (Statement)
(Statement)
NOT(Statement)
(Statement AND Statement)
(Statement IMPLY Statement)

Table 3: Statement Construction

Chapter 7: Logic-based Verification of Security Protocols

Logic-based verification of security protocols is a means of determining the correctness of a security protocol against a specified set of goals. Security protocols for mobile communication are chosen for verification and each protocol is detailed. A formal specification of the protocol messages is then formed using various formalization techniques. This specification is in the form of a text file that the parser in the CDVT verification engine reads. Along with the protocol messages, the initial assumptions of the protocol and the desired protocol goals are input to the CDVT engine following the textual conventions specified in the CDVT user guide [47].

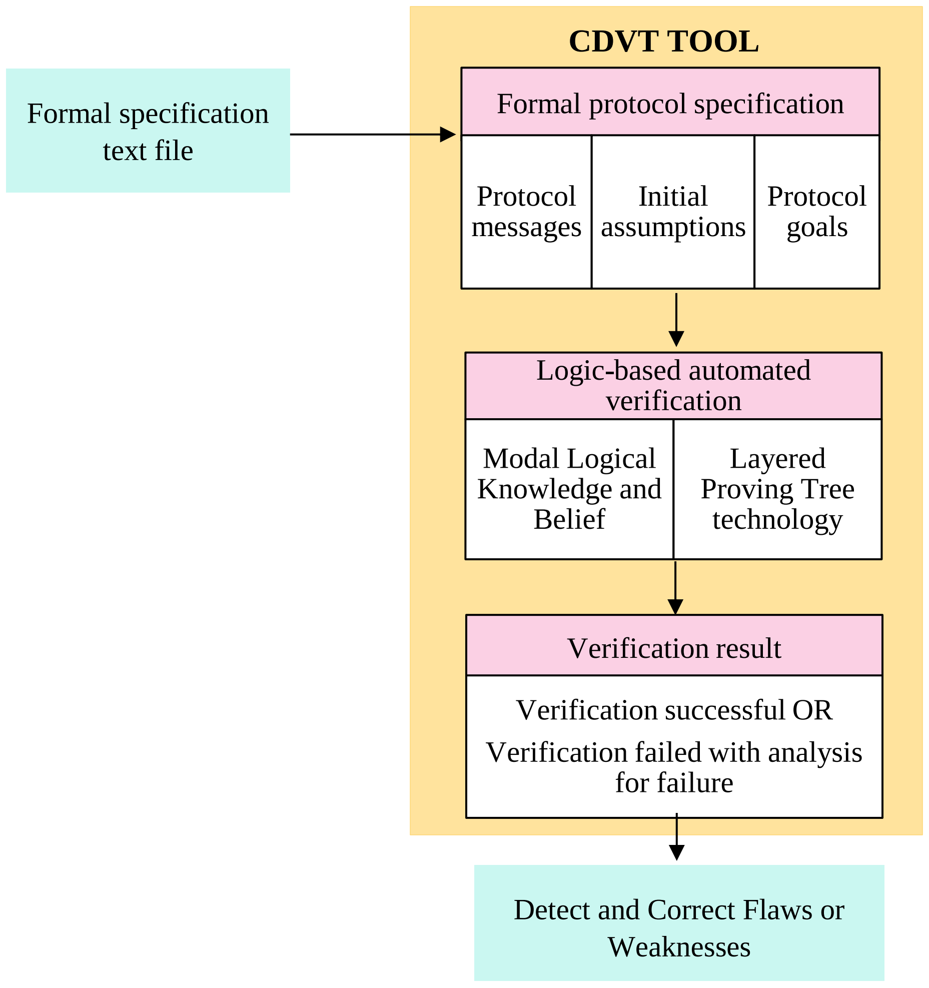

Formal specification text file
CDVT TOOL
Formal protocol specification
Protocol messages
Initial assumptions
Protocol goals
Logic-based automated verification
Modal Logical Knowledge and Belief
Layered Proving Tree technology
Verification result
Verification successful OR
Verification failed with analysis for failure
Detect and Correct Flaws or Weaknesses

Figure 10: Logic-based verification of security protocols

The tool then logically verifies the protocol specification against the desired goals by applying a set of axioms or postulates that have been previously defined in the tool. The result of the verification process is displayed as a visualization of a layered proving tree. If the verification is successful, the protocol is concluded to be flawless within the scope of the CDVT tool. Otherwise, the reason for failure is determined and the flaws or weaknesses are detected by re-tracing the tree and an attempt to correct these flaws are made.

The protocol verification output can reveal one of the following two results:

- **True** - if all the goals of the specified protocol are proven true. Hence, the verification is successful and the verified protocol can be considered secure within the scope of the CDVT Verification Logic.
- **False** – if at least one of the goals of the specified protocol is false (i.e. cannot be derived). Hence, the protocol verification fails and an investigation of the verification process can reflect weaknesses or flaws in the protocol design. If a weakness is discovered, the protocol should be re-designed and re-verified [3].

7.1 Project Description

The two major purposes of this project are to analyze the CDVT tool, and logically verify selected security protocols using this tool. The protocols to be employed are security protocols for wireless communication. The aim is to select recently published protocols for wireless communication and verify them with the tool. To familiarize with the tool, its input language is studied and specification files for some sample protocols are formulated in the language of the tool. The sample protocols used are Kao Chow Authentication v.1, Andrew Secure RPC, Needham-Schroeder Public Key, and Beller-Chang-Yacobi (BCY) protocol. These protocols are verified using the tool and the results noted. The results may reveal faults or weaknesses in the protocols as well as in the tool.

Formal specification files are then prepared for the selected security protocols for wireless communication, and these files are verified with the CDVT tool. The verification may disclose faults in the protocol which are noted. If the tool produces incorrect verification results, then it is considered as a fault in the tool and is recorded.

7.2 Verification of Sample Protocols

7.2.1 Formal Verification of Kao Chow Authentication v.1

The Kao Chow v.1 protocol [48] is a key distribution and nonce-based authentication protocol proposed by I-Lung Kao and Randy Chow claiming that using an uncertified key can give performance advantages and not necessarily reduce the security of authentication protocols, if the validity of the key can be verified at the end of an authentication process. The primary goal of this protocol is to achieve a low message overhead and strong authentication. Additionally, the protocol aims at preventing the compromise of session keys with repeated authentication.

7.2.1.1 Description of Kao Chow protocol

A and B are two principals wanting to authenticate each other and to obtain a shared session key for subsequent communication. The protocol uses a trusted third party - the authentication server, S - for key generation and distribution. Server S shares a master key with each principal – key Kas, Kbs with A and B respectively. Principals A and B trust S to issue a fresh secret session key, Kab which are then sent by S securely on the requests of principals. A and B will then use Kab to encrypt messages in future communication. In this protocol, nonce-based challenge/response exchanges are used to authenticate principals A and B to each other, and guarantee the freshness of messages.

The notations used to describe the Kao Chow Authentication v.1 protocol are as follows :

- A, B : principals
- S : principal (trusted third party – TTP in the formal specification)
- Kas : symmetric key shared by A,S
- Kbs : symmetric key shared by B,S
- Kab : fresh session key generated by S
- Na : nonce of A used in the authentication process
- Nb : nonce of B used in the authentication process
- {X}K : encryption of data X with the key K
- A -> B: Message : Principal A sends Message to B.

7.2.1.2 Informal specification of Kao Chow v.1 protocol

The following are the steps of the Kao Chow Authentication v.1 :

1. A -> S : A, B, Na
2. S -> B : {A, B, Na, Kab}Kas, {A, B, Na, Kab}Kbs
3. B -> A : {A, B, Na, Kab}Kas, {Na}Kab, Nb
4. A -> B : {Nb}Kab

In the first step, A sends to trusted server S its identity, B’s identity and a fresh nonce Na.

In the second step, S sends two messages to B, each containing the identities of A and B, A’s nonce Na, and a secret session key generated by S, Kab. The first message S sends is encrypted using a symmetric key shared by A and S, Kas. The second message is encrypted using a symmetric key shared between B and S. B then decrypts the second message and obtains the session key, Kab.

In the third step, B forwards to A the first message it has obtained from S in the second step, along with a fresh nonce, Nb, and A’s nonce Na encrypted using Kab. A obtains the session key, Kab by decrypting the message forwarded to it by B, and then decrypts the second message to obtain nonce Na to verify the correctness of the encryption.

In the last step, A sends to B nonce Nb encrypted using key Kab. On reception of this message, B verifies the correctness of the encrypted nonce Nb.

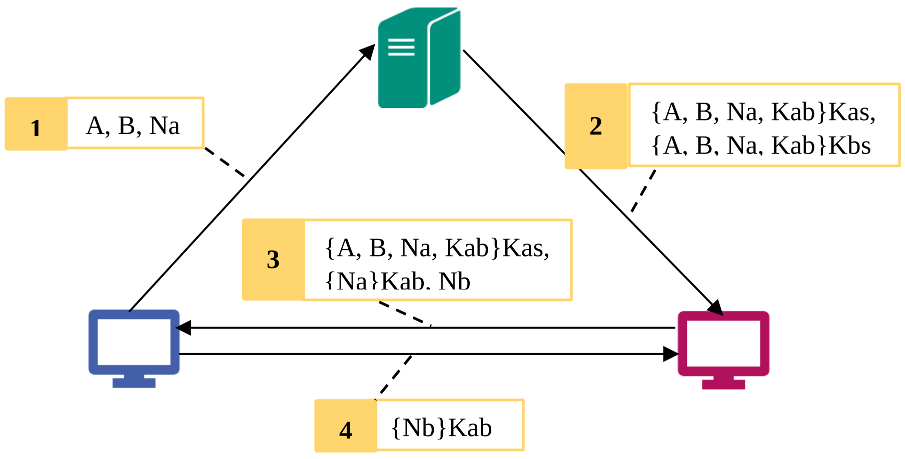


Figure 11: Kao Chow Authentication v.1

7.2.1.3 Formal Specification of Kao Chow v.1 protocol

The formal specification of a protocol includes three parts :

1. Initial Assumptions
2. Protocol Steps
3. Protocol Goals

- **Initial Assumptions**

Initial assumptions are statements defining what each principal possesses and knows at the beginning of a protocol run. The following are the initial assumptions of the Kao Chow v.1 protocol (Server S, here is represented as TTP) :

```
A1:  A possess at[0] Kas;
A2:  A know at[0] TTP possess at[0] Kas;
A3:  A possess at[0] Na;
A4:  A know at[0] NOT(Zero possess at[0] Na);
A5:  B possess at[0] Kbs;
A6:  B know at[0] TTP possess at[0] Kbs;
A7:  B possess at[0] Nb;
A8:  B know at[0] NOT(Zero possess at[0] Nb);
A9:  TTP possess at[0] Kas;
A10: TTP possess at[0] Kbs;
A11: TTP possess at[0] Kab;
A12: TTP know at[0] NOT(Zero possess at[0] Kab);
A13: A know at[3] ( ((A receivefrom B at[3] {Na}Kab AND A know at
     [0] NOT(Zero possess at[0] {Na}Kab)) AND A know at[3] TTP
     send at[3] {A,B,Na,Kab}Kas) IMPLY (B possess at[3] Kab) );
A14: B know at[0] (B know at[2] TTP send at[2] {A,B,Na,Kab}Kbs
     AND B know at[2] NOT(Zero send at[0] {A,B,Na,Kab}Kbs ) )
     IMPLY A possess at[4] Kab);
```

The first assumption, A1, represents the fact that principal A possesses key Kas at the beginning of the protocol run. A2 states that A knows before the beginning of the protocol run, that TTP also possesses key Kas at the same time. A3 expresses that A possesses nonce Na. Assumption A4 represents the fact that before start of the protocol run, principal A is aware that no other principal possesses this nonce at that time.

Similarly, A5 represents the fact that principal B possesses key Kbs at the beginning of the protocol run. A6 expresses that B knows before the beginning of the protocol run, that TTP also possesses key Kbs at the same time. A7 states that B possesses nonce Nb. Assumption A8 represents the fact that before start of the protocol run, principal B is aware that no other principal possesses this nonce at that time.

Assumptions A9, A10 and A11 states that TTP possesses Kas, Kbs and secret key Kab, respectively at the beginning of the protocol run. A12 represents the fact that before start of the protocol run, TTP is aware that no other principal possesses secret key Kab at that time.

A13 states that A receives from B {Na}Kab and no other principal possesses {Na}Kab to prove the authenticity of B, and that if principal A can deduce that the message {A,B,Na,Kab}Kas has been sent by TTP and that it has been sent during the current protocol run, then A can also deduce that Kab is a good session key shared with B and that B possesses this session key.

Similarly, A14 states that if principal B can deduce that the message {A,B,Na,Kab}Kbs has been sent by TTP and also that it has been send during the current protocol run, then B can also deduce that Kab is a good session key shared with A and that A possesses this session key.

- **Protocol Steps**

The Kao Chow v.1 protocol steps are formalized as follows:

```
S1:   TTP receivefrom A at[1] A,B,Na;
S2:   B receivefrom TTP at[2] {A,B,Na,Kab}Kas, {A,B,Na,Kab}Kbs;
S3:   A receivefrom B at[3] {A,B,Na,Kab}Kas, {Na}Kab, Nb;
S4:   B receivefrom A at[4] {Nb}Kab;
```

Statement S1 states that TTP receives from principal A in step one (at time t1) a request, which contains A's identity, B's identity (the principal A wants to communicate to) and a nonce, Na.

S2 states that B receives from TTP in step two (at time t2) two tickets – the first one containing identities of A and B, nonce Na, secret key Kab generated by TTP, all encrypted using Kas (shared key between A and TTP); and the second one containing identities of A and B, nonce Na, secret key Kab generated by TTP, all encrypted using Kbs (shared key between B and TTP).

S3 states that A receives from B in step three (at time t3) the first ticket B received from TTP, the nonce Na encrypted using key Kab, and its own nonce, Nb.

S4 states that B receives from A in step four (at time t4) nonce Nb encrypted with key Kab.

- **Protocol Goals**

The objectives of Kao Chow v.1 protocol are the mutual authentication of principals A and B and the distribution and secrecy of the session key, Kab.

These objectives are specified in the goals as follows :

```
G1: B possess at[2] Kab;
G2: B know at[2] TTP send at[2] {A,B,Na,Kab}Kbs;
G3: B know at[2] NOT(Zero send at[0] {A,B,Na,Kab}Kbs);
G4: A possess at[3] Kab;
G5: A know at[3] TTP send at[3] {A,B,Na,Kab}Kas;
G6: A know at[3] NOT(Zero send at[0] {A,B,Na,Kab}Kas);
G7: A know at[3] B send at[3] {Na}Kab;
G8: A know at[3] NOT(Zero send at[0] {Na}Kab);
G9: B know at[4] A send at[4] {Nb}Kab;
G10: B know at[4] NOT(Zero send at[0] {Nb}Kab);
```

Goal G1 states that B possesses the session key Kab after step 2 is completed and G2 states that B knows that the message that contains the session key originates at TTP. Goal G3 states that B knows that this message is fresh, i.e. it has been created by TTP for the current protocol run. The above three goals represent key establishment for B.

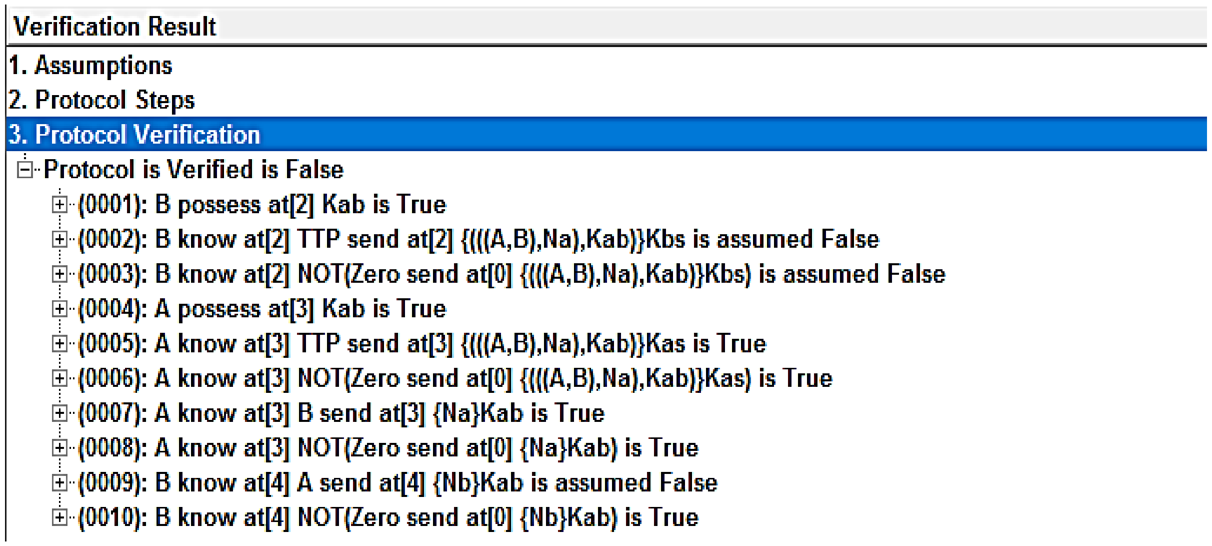
G4 states that A possesses the session key Kab after step 3 is completed and G5 states that A knows that the message that contains the session key originates at TTP. Goal G6 states that A knows that this message is fresh, i.e. it has been created by TTP for the current protocol run. The above three goals represent key establishment for A.

G7 states that A knows B is certainly the source of message {Na}Kab, which is the response to the challenge of A's nonce. G8 states that A knows that this message has been created during the current protocol run. Goals G7 and G8 represent authentication of principal B to A.

G9 states that B knows A is certainly the source of message {Nb}Kab, which is the response to the challenge of B's nonce. G10 states that A knows that this message has been created during the current protocol run. Goals G9-G10 are goals corresponding authentication of A to B.

7.2.1.4 Verification Results of Kao Chow v.1 protocol

The result for the formal verification of Kao Chow Authentication v.1 using the CDVT tool is shown in the following figure.



```
Verification Result
1. Assumptions
2. Protocol Steps
3. Protocol Verification
  Protocol is Verified is False
    (0001): B possess at[2] Kab is True
    (0002): B know at[2] TTP send at[2] {(((A,B),Na),Kab)}Kbs is assumed False
    (0003): B know at[2] NOT(Zero send at[0] {(((A,B),Na),Kab)}Kbs) is assumed False
    (0004): A possess at[3] Kab is True
    (0005): A know at[3] TTP send at[3] {(((A,B),Na),Kab)}Kas is True
    (0006): A know at[3] NOT(Zero send at[0] {(((A,B),Na),Kab)}Kas) is True
    (0007): A know at[3] B send at[3] {Na}Kab is True
    (0008): A know at[3] NOT(Zero send at[0] {Na}Kab) is True
    (0009): B know at[4] A send at[4] {Nb}Kab is assumed False
    (0010): B know at[4] NOT(Zero send at[0] {Nb}Kab) is True
```

Figure 12: Verification Results for Kao Chow v.1

It is clear from the above verification result that the goals relating to key establishment for A and the authentication of B to A are verified as True whereas the goals relating to key establishment for B and the authentication of A to B are verified as False. The verification has failed due to B’s inability to establish A’s possession of the session key Kab.

Exploring further into the failed protocol goals, reveals that the protocol suffers from a freshness weakness. This weakness is due to B’s inability to establish freshness of the message component containing the session key in protocol step 2. This prevents B to accept the session key. Therefore, goal G9 also fails. Thus, neither key establishment for B nor authentication of A to B is achieved by the protocol, and hence, the protocol is verified as False.

7.2.2 Formal Verification of Andrew Secure RPC protocol

7.2.2.1 Description of Andrew Secure RPC

The Andrew Secure Remote Procedure Call (RPC) is a protocol that allows two parties to establish a new cryptographic key, The protocol was originally designed to work on the Andrew network of Carnegie Mellon University, USA. Andrew is a distributed computing environment that has been under development at Carnegie Mellon University since 1983.

Andrew Secure RPC protocol allows two communicating parties, A and B (usually, a client and a server), who already share a key Kab, to agree upon a new key Kab1. The protocol also performs an authentication handshake. There are four messages in the protocol exchange. The first three, A and B perform a handshake using a shared secret Kab. In the final message, B sends a new key Kab1 to A.

The notations used to describe the Andrew Secure RPC protocol are as follows :

- A, B : principals
- Kab, Kab1 : symmetric keys
- Na : nonce of A
- Nb, Nb1 : nonces of B
- succ : nonce -> nonce

The following are the steps of the protocol :

1. A -> B : A, {Na}Kab
2. B -> A : {succNa, Nb}Kab
3. A -> B : {succNb}Kab
4. B -> A : {Kab1, Nb1}Kab

7.2.2.2 Verification Results of Andrew Secure RPC

The Andrew Secure RPC protocol is verified as False using the CDVT tool, indicating that the protocol has a few faults or weaknesses. Exploring the faults further has produced the following weaknesses:

- A’s inability to establish that B possesses at step 4 the newly generated key Kab1
- B’s inability to establish that A possesses at step 4, Kab1
- A’s inability to establish that message in step 4 is fresh indicates that Kab1 might also not be fresh.
- Failure of goal G4 indicate that key Kab1 might be compromised.

Thus, there is no assurance that key Kab1 is fresh and is not substituted from a previously recorded message 4, and therefore the protocol does not hold.

7.2.3 Formal Verification of Needham-Schroeder Public Key protocol

7.2.3.1 Description of Needham-Schroeder Public Key protocol

The Needham–Schroeder Public-Key Protocol, based on public-key cryptography. This protocol is intended to provide mutual authentication between two parties communicating on a network. In this protocol, two principals A and B use a trusted server S (trusted third party, TTP in the CDVT tool) to distribute public keys on request.

The notations used to describe the Andrew Secure RPC protocol are as follows :

- A, B, S : Principals
- Na, Nb : Nonces

- KPa, KPb, KPs, KSa, KSb, KSs : Keys
- KPa, KSa : is a key pair
- KPb, KSb : is a key pair
- KPs, KSs : is a key pair

The following are the steps of the protocol :

1. A -> S : A,B
2. S -> A : {KPb, B}KSs
3. A -> B : {Na, A}KPb
4. B -> S : B,A
5. S -> B : {KPa, A}KSs
6. B -> A : {Na, Nb}KPa
7. A -> B : {Nb}KPb

7.2.3.2 Verification Results of Needham-Schroeder Public Key protocol

The Needham-Schroeder Public Key protocol is verified as False by the CDVT tool. The reason for the failure of the verification is that the protocol fails to protect against a man-in-the-middle attack where an intruder can replay the messages.

The protocol fails to provide mutual authentication between the two principals A and B and it is evident from the following results:

- A's inability to establish that B sent the message {Na,Nb}KaPub in step S6.
- B's inability to establish that A sent the message {Nb}KbPub in step S7.
- A's and B's inability to establish that messages in step S2 and step S4 respectively, are fresh indicates that the keys also might not be fresh.

7.2.4 Formal Verification of Beller-Chang-Yacobi protocol

7.2.4.1 Description of BCY protocol

The BCY protocol [49] is designed to satisfy the requirements of the mobile communications environment. They are intended to provide security between a mobile station and a base station of the fixed network. The BCY protocol aims to provide authentication and key agreement for low-power portable devices such as mobile phones. Due to low computational power required, BCY protocol demonstrates the computational feasibility of using public-key cryptography in mobile communications, by combining a Diffie-Hellman key exchange with the modular square root (MSR) encryption technique.

The notations used to describe the BCY protocol are as follows :

- U : An identifier of the user
- V : An identifier of the service provider
- S : An identifier of the certification authority
- Kvd+ : V's public key for Diffie–Hellman key agreement
- Kvm+ : V's public key for MSR encryption
- KK : A key-encrypting key
- SK : A session key
- rU : A random nonce generated by U

The following are the steps of the protocol :

1. V -> U : {V, Kvd+,Kvm+}Ks-

   U computes Y = {rU}Kvm+, KK={Kvd+}Ku-, SK = {rU}KK
2. U -> V : Y, {{U,Ku+}Ks-}rU

   V computes rU ={Y}Kvm-, KK={Ku+}Kvd-, SK = {rU}KK
3. V -> U : {dataV}SK
4. U -> V : {dataU}SK

7.2.4.2 Verification Results of BCY protocol

The automated verification of the BCY protocol identifies the following failed goals:

- U’s inability to establish that V’s public keys are valid as the validity of the certificates cannot be established, and hence, authentication fails.
- V’s inability to establish that U’s public key is valid as the validity of the certificates cannot be established, and hence, authentication fails.
- V’s inability to establish the knowledge of freshness of rU nonce.
- U’s inability to establish that the session key is a suitable shared secret with V and hence, V cannot establish that the session key is fresh.

7.3 Conclusion

This chapter details the formal verification of sample protocols using the CDVT tool. The verification results obtained refer to weaknesses in the protocol or in the tool. In the above verification results, weaknesses have been found in the protocols and hence, they have been verified as False by the CDVT tool. Therefore, the task of familiarizing with the tool and the language has been completed. The next task is to select recently proposed security protocols for wireless communication, form formal specification files for these protocols in the language of the tool, and verify the protocols using CDVT.

Chapter 8: Logic-based Verification of Security Protocols for Mobile/Wireless communication using CDVT

8.1 Introduction

This chapter includes formal specifications of eight recent security protocols for wireless communication along with their verification results obtained from the CDVT tool. There are three protocols published in the year 2017, four protocols published in 2016, and one published in 2015. This chapter contains sections containing each one of these protocols, and sub-sections consisting of their formal specifications (assumptions, steps and goals), and their formal verification results in CDVT.

The security protocols for wireless communication mentioned in the following sections are:

1. Classiquantum Resistance Secure Simple Pairing (CRSSP)
2. HIP-based Secure Software Defined Mobile Networks (SDMN)
3. Secure Mobile Wallet
4. Secure patient-health monitoring Wireless Body Area Networks (WBAN)
5. P2P service security protocol in M2M environment
6. Secure communication in vehicular networks
7. TTP based High-Efficient Muti-key Exchange Protocol (THMEP)
8. MAKA scheme with user anonymity in GLOMONET

8.2 Classiquantum Resistance Secure Simple Pairing (CRSSP)

The CRSSP [12] protocol has been developed to overcome the limitations of the initial Secure Simply Pairing(SSP) protocol by making it safe to use in classical as well as quantum computers. The security of all cryptographic systems depends on the hardness of the technique used to compute the key. CRSSP uses Ring learning with errors (RLWE) keys, currently used in public-key cryptography, to protect against cryptanalytic attacks in quantum computers.

The notations used in the formal specification of CRSSP [12]protocol are as follows:

- A, B : Principals
- KaPub1, KaPub2 : RLWE Public keys of A
- KaPriv1, KaPriv2 : RLWE Private keys of A
- KbPub1, KbPub2 : RLWE Public keys of B
- KbPriv1, KbPriv2 : RLWE Private keys of B
- ioCapA, ioCapB : IO capabilities of A and B
- bdAddressA, bdAddressB : Bluetooth Device Addresses
- Na, Nb : Nonces

The following are the steps of the protocol:

1. A -> B : bdAddressA, KaPub1

   B : SecretKeyB1 = F(KbPriv1, KbPub1, KaPub1)
2. B -> A : bdAddressB, KbPub1

   A : SecretKeyA1 = F(KaPriv1, KaPub1, KbPub1)
3. A -> B : Na

   B : SignatureB = H(SecretKeyB1, Nb, Na, KbPub1, KaPub1)
4. B -> A : Nb

   A : SignatureA = H(SecretKeyA1, Na, Nb, KaPub1, KbPub1)
5. A -> B : SignatureA
6. B -> A : SignatureB
7. A -> B : KaPub2

   B : SecretKeyB2 = F(KbPriv2, KbPub2, KaPub2)

   SessionKeyB = H(SecretKeyB1,SecretKeyB2, ioCapB, bdAddressB, KbPub2, KaPub2)
8. B -> A : KbPub2

   A : SecretKeyA2 = F(KaPriv2, KaPub2, KbPub2)

   SessionKeyA = H(SecretKeyA1,SecretKeyA2,ioCapA, bdAddressA, KaPub2,KbPub2)

9. A -> B : {dataA}SessionKeyA
10. B -> A : {databB}SessionKeyB

8.2.1 Formal Verification of CRSSP protocol

- **Initial Assumptions**

Initial assumptions are statements defining what each principal possesses and knows at the beginning of a protocol run.

The initial assumptions for CRSSP protocol is as follows:

```
A1: A possess at[0] KaPub1;
A2: A possess at[0] KaPriv1;
A3: A possess at[0] KaPub2;
A4: A possess at[0] KaPriv2;
A5: A possess at[0] dataA;
A6: A possess at[0] bdAddressA;
A7: A possess at[0] ioCapA;
A8: A possess at[0] Na;
A9: A know at[0] NOT(Zero possess at[0] Na);
A10: A know at[0] B possess at[0] KbPriv1;
A11: A know at[0] B possess at[0] KbPriv2;
A12: A know at[0] NOT(Zero possess at[0] dataB);
A13: B possess at[0] KbPub1;
A14: B possess at[0] KbPriv1;
A15: B possess at[0] KbPub2;
A16: B possess at[0] KbPriv2;
A17: B possess at[0] dataB;
A18: B possess at[0] bdAddressB;
A19: B possess at[0] ioCapB;
A20: B possess at[0] Nb;
A21: B know at[0] NOT(Zero possess at[0] Nb);
```

```
A22: B know at[0] A possess at[0] KaPriv1;
A23: B know at[0] A possess at[0] KaPriv2;
A24: B know at[0] NOT(Zero possess at[0] dataA);
A25: B know at[7] (((B receivefrom A at[7] {dataA}H(F(KaPriv1,
     KaPub1, KbPub1), F(KaPriv2, KaPub2, KbPub2), ioCapA,
     bdAddressA, KaPub2, KbPub2) AND B know at[0] A possess at[0]
     KaPriv1) AND B know at[0] A possess at[0] KaPriv2) IMPLY (A
     send at[7] {dataA}H(F(KaPriv1, KaPub1, KbPub1), F(KaPriv2,
     KaPub2, KbPub2), ioCapA, bdAddressA, KaPub2, KbPub2)));
A26: A know at[8] (((A receivefrom B at[8] {dataB}H(F(KbPriv1,
     KbPub1, KaPub1), F(KbPriv2, KbPub2, KaPub2), ioCapB,
     bdAddressB, KbPub2, KaPub2) AND A know at[0] B possess at[0]
     KbPriv1) AND A know at[0] B possess at[0] KbPriv2) IMPLY (B
     send at[8] {dataB}H(F(KbPriv1, KbPub1, KaPub1), F(KbPriv2,
     KbPub2, KaPub2), ioCapB, bdAddressB, KbPub2, KaPub2)));
```

Assumptions A1, A2, A3, A4, A5, A6, A7 and A8 states that principal A possesses KaPub1, KaPriv1, KaPub2, KaPriv2, dataA, bdAddressA, ioCapA, and Na respectively. Assumption A9 represents the fact that before start of the protocol run, principal A is aware that no other principal possesses this nonce Na at that time. A10 and A11 states that A knows principal B possesses KbPriv1 and KbPriv2 before the start of the protocol run.

Similarly, assumptions A13, A14, A15, A16, A17, A18, A19 and A20 represents the fact that principal B possesses KbPub1, KbPriv1, KbPub2, KbPriv2, dataB, bdAddressB, ioCapB, and Nb respectively, at the beginning of the protocol run. Assumption A21 represents the fact that before start of the protocol run, principal B is aware that no other principal possesses this nonce Nb at that time. A22 and A23 expresses that B knows before the beginning of the protocol run, that A possesses keys KaPriv1, KaPriv2 at the same time.

Assumption A25 implies that if B know only A possesses KaPriv1 and B has received from A certain data containing KaPriv1, then B knows it is A that has sent the data. A26 implies that if A know only B possesses KbPriv1 and A has received from B certain data containing KbPriv1, then A knows it is B that has sent the data.

- **Protocol Steps**

The CRSSP protocol steps are formalized as follows:

```
S1: B receivefrom A at[1] ioCapA, bdAddressA, KaPub1, Na;
S2: A receivefrom B at[2] ioCapB, bdAddressB, KbPub1, Nb;
S3: B receivefrom A at[3] H(F(KaPriv1, KaPub1, KbPub1), Na, Nb,
    KaPub1, KbPub1);
S4: A receivefrom B at[4] H(F(KbPriv1, KbPub1, KaPub1), Nb, Na,
    KbPub1, KaPub1);
S5: B receivefrom A at[5] KaPub2;
S6: A receivefrom B at[6] KbPub2;
S7: B receivefrom A at[7] {dataA}H(F(KaPriv1, KaPub1, KbPub1),
    F(KaPriv2, KaPub2, KbPub2), ioCapA, bdAddressA, KaPub2,
    KbPub2);
S8: A receivefrom B at[8] {dataB}H(F(KbPriv1, KbPub1, KaPub1),
    F(KbPriv2, KbPub2, KaPub2), ioCapB, bdAddressB, KbPub2,
    KaPub2);
```

Statement S1 states that B receives from principal A in step one (at time t1) a message, which contains A’s IO capability, A’s Bluetooth device address, A’s public key and a nonce, Na. S2 states that A receives from B in step two (at time t2) a message, which contains B’s IO capability, B’s Bluetooth device address, B’s public key and a nonce, Nb.

S3 states B receives from A at time t3 hash of a function containing A’s public and private key and B’s public key, nonce of A and B, public keys of A and B. Similarly, in S4, A receives from B at time t4 hash of a function containing B’s public and private key and A’s public key, nonce of B and A, public keys of B and A respectively. S5 states B receives from A the second public key of A and S6 states A receives from B the second public key of B.

S7 states that B receives from A dataA encrypted with a key that contains a hash of – a function containing the first public and private keys of A and first public key of B, another function with second public and private keys of A and second public key of B, IO capability and device address of A and second public key of A and B. Similarly, S8 states A receives from B dataB encrypted with a key that contains a hash of – a function containing the first public and private keys of B and first public key of A, another function with second public and private keys of B and second public key of A, IO capability and device address of B and second public key of B and A respectively.

- **Protocol Goals**

The major objective of CRSSP protocol is to increase the security of pairing and authentication between Bluetooth devices for communication.

These objectives are specified in the goals as follows:

```
G1: B possess at[1] KaPub1;
G2: A possess at[2] KbPub1;
G3: B possess at[3] H(F(KaPriv1, KaPub1, KbPub1), Na, Nb, KaPub1,
    KbPub1);
G4: B know at[3] (not(Zero possess at[0] H(F(KaPriv1, KaPub1,
    KbPub1), Na, Nb, KaPub1, KbPub1)));
G5: A possess at[4] H(F(KbPriv1, KbPub1, KaPub1), Nb, Na, KbPub1,
    KaPub1);
G6: A know at[4] (not(Zero possess at[0] H(F(KbPriv1, KbPub1,
    KaPub1), Nb, Na, KbPub1, KaPub1)));
G7: B possess at[5] KaPub2;
G8: A possess at[6] KbPub2;
G9: A possess at[7] {dataA}H(F(KaPriv1, KaPub1, KbPub1),
    F(KaPriv2, KaPub2, KbPub2), bdAddressA, KaPub2, KbPub2);
G10: B know at[7] not(Zero possess at[0] {dataA}H(F(KaPriv1,
     KaPub1, KbPub1), F(KaPriv2, KaPub2, KbPub2), ioCapA,
     bdAddressA, KaPub2, KbPub2));
```

```
G11: B know at[7] A send at[7] {dataA}H(F(KaPriv1, KaPub1,
     KbPub1), F(KaPriv2, KaPub2, KbPub2), ioCapA, bdAddressA,
     KaPub2, KbPub2);
G12: B possess at[8] {dataB}H(F(KbPriv1, KbPub1, KaPub1),
     F(KbPriv2, KbPub2, KaPub2), ioCapB, bdAddressB, KbPub2,
     KaPub2);
G13: A know at[8] not(Zero possess at[0] {dataB}H(F(KbPriv1,
     KbPub1, KaPub1), F(KbPriv2, KbPub2, KaPub2), ioCapB,
     bdAddressB, KbPub2, KaPub2));
G14: A know at[8] B send at[8] {dataB}H(F(KbPriv1, KbPub1,
     KaPub1), F(KbPriv2, KbPub2, KaPub2), ioCapB, bdAddressB,
     KbPub2, KaPub2);
```

Goal G1 states that B possesses KaPub1 at step 1 and G2 states that A possesses KbPub1 after step 2 is completed. Goal G3 states that B possesses H(F(KaPriv1, KaPub1, KbPub1), Na, Nb, KaPub1, KbPub1) at step 3 and G4 states that B knows that this message is fresh, i.e. it has been created by A for the current protocol run. G5 states that A possesses H(F(KbPriv1, KbPub1, KaPub1), Nb, Na, KbPub1, KaPub1) after step 4 is completed and G6 states that A knows that this message is fresh, i.e. it has been created by B for the current protocol run.

Goal G7 states that B possesses KaPub2 at step 5 and G8 states that A possesses KbPub2 after step 6 is completed. G9 states that A possess after step 7 {dataA}H(F(KaPriv1, KaPub1, KbPub1), F(KaPriv2, KaPub2, KbPub2), bdAddressA, KaPub2, KbPub2) and G10 states B knows that this message is fresh and has been created during the current protocol run. G11 states that B knows A is certainly the source of this message. The above steps represent authentication of A to B.

Goal G12 states B possess after step 8 {dataB}H(F(KbPriv1, KbPub1, KaPub1), F(KbPriv2, KbPub2, KaPub2), ioCapB, bdAddressB, KbPub2, KaPub2) and G13 states A knows that this message is fresh and has been created during the current protocol run. G14 states that A knows B is certainly the source of this message. The above steps represent authentication of B to A.

8.2.2 Verification results of CRSSP in CDVT

The verification results of CRSSP protocol represents that the protocol is true, i.e, it satisfies all the goals and the tool found no weaknesses in the protocol.

Goals G9 to G11 are verified as true, leading to the fact that the authentication of principal A to B is successful. Similarly, goals G12 to G14 are verified as true, proving that the authentication of principal B to A is successful. This protocol is safe against replay attacks as all the previously generated public keys, private keys, nonces and link/session keys are discarded immediately after use and new/fresh values for these objects are computed in next session.

CRSSP [12] also resists man-in-the-middle (MITM) attack as it is a strong quantum resistant pairing protocol and it is extremely difficult even for quantum computers - with increasing memory, speed, and decreasing computation time of CPU - to compute the lattice based R-LWE shared secret key from the public key. It is hard to compute the link/session key since the link key is strongly encapsulated by two R-LWE based secret keys and HMAC-SHA256 function. Therefore, CRSSP withstands MITM attack in classical as well as quantum computers.

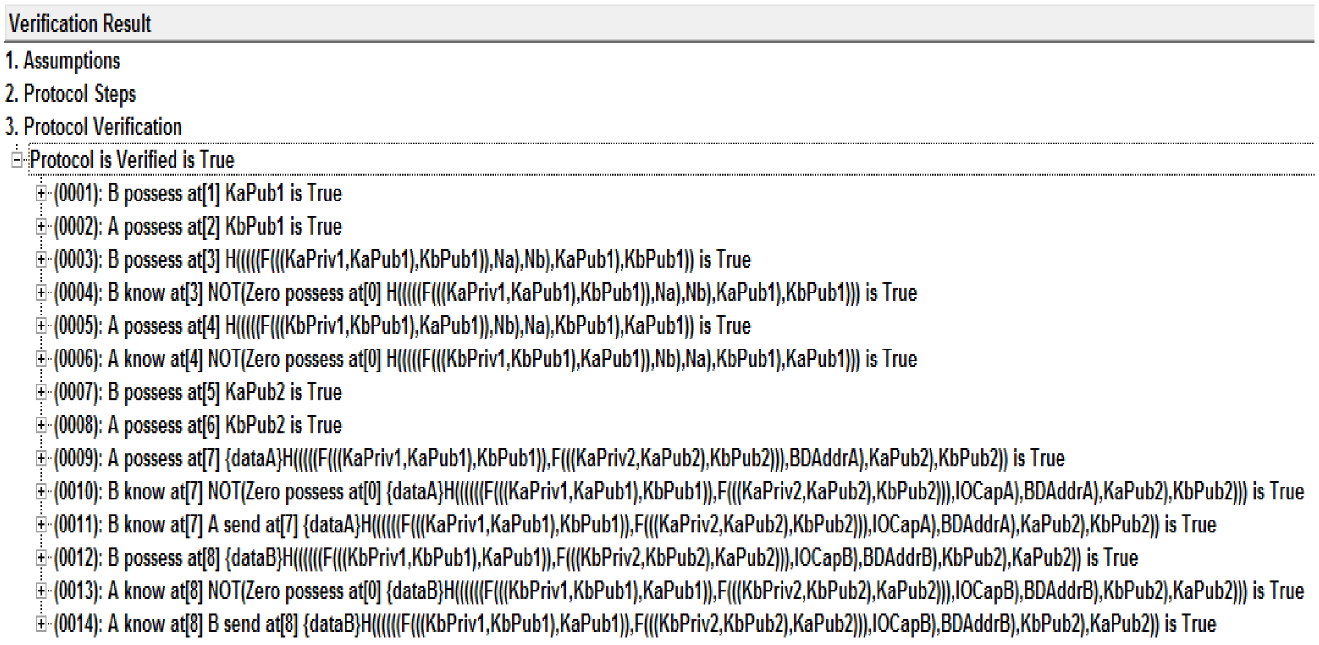

```
Verification Result
1. Assumptions
2. Protocol Steps
3. Protocol Verification
 Protocol is Verified is True
   (0001): B possess at[1] KaPub1 is True
   (0002): A possess at[2] KbPub1 is True
   (0003): B possess at[3] H(((((F(((KaPriv1,KaPub1),KbPub1)),Na),Nb),KaPub1),KbPub1)) is True
   (0004): B know at[3] NOT(Zero possess at[0] H(((((F(((KaPriv1,KaPub1),KbPub1)),Na),Nb),KaPub1),KbPub1))) is True
   (0005): A possess at[4] H(((((F(((KbPriv1,KbPub1),KaPub1)),Nb),Na),KbPub1),KaPub1)) is True
   (0006): A know at[4] NOT(Zero possess at[0] H(((((F(((KbPriv1,KbPub1),KaPub1)),Nb),Na),KbPub1),KaPub1))) is True
   (0007): B possess at[5] KaPub2 is True
   (0008): A possess at[6] KbPub2 is True
   (0009): A possess at[7] {dataA}H(((((F(((KaPriv1,KaPub1),KbPub1)),F(((KaPriv2,KaPub2),KbPub2))),BDAddrA),KaPub2),KbPub2)) is True
   (0010): B know at[7] NOT(Zero possess at[0] {dataA}H((((((F(((KaPriv1,KaPub1),KbPub1)),F(((KaPriv2,KaPub2),KbPub2))),IOCapA),BDAddrA),KaPub2),KbPub2))) is True
   (0011): B know at[7] A send at[7] {dataA}H((((((F(((KaPriv1,KaPub1),KbPub1)),F(((KaPriv2,KaPub2),KbPub2))),IOCapA),BDAddrA),KaPub2),KbPub2)) is True
   (0012): B possess at[8] {dataB}H((((((F(((KbPriv1,KbPub1),KaPub1)),F(((KbPriv2,KbPub2),KaPub2))),IOCapB),BDAddrB),KbPub2),KaPub2)) is True
   (0013): A know at[8] NOT(Zero possess at[0] {dataB}H((((((F(((KbPriv1,KbPub1),KaPub1)),F(((KbPriv2,KbPub2),KaPub2))),IOCapB),BDAddrB),KbPub2),KaPub2))) is True
   (0014): A know at[8] B send at[8] {dataB}H((((((F(((KbPriv1,KbPub1),KaPub1)),F(((KbPriv2,KbPub2),KaPub2))),IOCapB),BDAddrB),KbPub2),KaPub2)) is True
```

Figure 13: Verification Result of CRSSP in CDVT

8.3 HIP-based Secure Software Defined Mobile Networks (SDMN)

Host Identity Protocol (HIP) is a mobility and security management protocol which is standardized by IETF. HIP separates the dual role of IP address as the locater and the host identity. Each HIP host has a public/private key pair and the public key is used as its Host Identity (HI). HIP utilizes a base protocol, HIP Base Exchange (HIP BEX), to mutually authenticate the end nodes. HIP establishes a Security Association (SA) for IPsec tunnels.

The two communication channels for SDMN [13] are control channel and data channel.

8.3.1 HIP-based control channel

The following are the notations used in the formal specification of the authentication and tunnel establishment procedure of control channel [13]:

- DPS : Data Plane Switch (A)
- SecGW : Security Gateway(B)
- Nseq : Unique session identifier
- Nb : Nonce of SecGW(B)
- Kab : Shared Key
- KaPub, KbPub : Public keys of DPS(A) and SecGW(B) respectively
- KaPriv, KbPriv : Private keys of DPS(A) and SecGW(B) respectively

The following are the steps of the protocol:

1. DPS -> SecGW : H(KaPub), H(KbPub), Nseq
2. SecGW -> DPS : H(KbPub), H(KaPub), { Nb, {Kab}KaPub, KbPub, Nseq, data_echo_Request }KbPriv
3. DPS -> SecGW : H(KaPub), H(KbPub), { F(Nb), {Kab}KbPub, KaPub, Nseq, data_echo_Response }KaPriv

8.3.2 Formal Verification of HIP-based control channel

- **Initial Assumptions**

Initial assumptions are statements defining what each principal possesses and knows at the beginning of a protocol run.

The initial assumptions for HIP-based control channel of SDMN is as follows:

```
A1: DPS possess at[0] KaPub;
A2: DPS possess at[0] KaPriv;
A3: DPS possess at[0] KbPub;
A4: DPS possess at[0] Nseq;
A5: DPS know at[0] NOT(Zero possess at[0] Nseq);
A6: DPS know at[0] SecGW possess at[0] KbPriv;
A7: DPS know at[0] KMaterial(Kab);
A8: SecGW possess at[0] KbPub;
A9: SecGW possess at[0] KbPriv;
A10: SecGW possess at[0] KaPub;
A11: SecGW possess at[0] Nb;
A12: SecGW know at[0] NOT(Zero possess at[0] Nb);
A13: SecGW know at[0] DPS possess at[0] KaPriv;
A14: SecGW know at[0] KMaterial(Kab);
```

Assumptions A1, A2, A3 and A4 states that principal DPS possesses KaPub, KaPriv, KbPub and Nseq respectively. Assumption A5 represents the fact that before start of the protocol run, principal DPS is aware that no other principal possesses this nonce Nseq at that time. A6 states that DPS knows principal SecGW possesses KbPriv before the start of the protocol run. A7 states that DPS knows before the start of the protocol run, that key Kab is a shared key between principals DPS and SecGW.

Similarly, assumptions A8, A9, A10 and A11 states that principal SecGW possesses KbPub, KbPriv, KaPub and Nb respectively. Assumption A12 represents the fact that before start of the protocol run, principal SecGW is aware that no other principal possesses this nonce Nb at that time.

A13 states that SecGW knows principal DPS possesses KaPriv before the start of the protocol run. A14 states that SecGW knows before the start of the protocol run, that key Kab is a shared key between principals DPS and SecGW.

- **Protocol Steps**

The HIP-based secure SDMN control channel protocol steps are formalized as follows:

```
S1: SecGW receivefrom DPS at[1] H(KaPub), H(KbPub), Nseq;
S2: DPS receivefrom SecGW at[2] H(KbPub), H(KaPub), {Nb,
    {Kab}KaPub, KbPub, Nseq, data_echo_Request}KbPriv;
S3: SecGW receivefrom DPS at[3] H(KaPub), H(KbPub), {F(Nb),
    {Kab}KbPub, KaPub, Nseq, data_echo_Response}KaPriv;
```

Statement S1 states that SecGW receives from principal DPS in step one (at time t1) a message, which contains hash of DPS's public key, hash of SecGW's public key and a nonce, Nseq. S2 states that DPS receives from SecGW in step two (at time t2) a message, which contains hash of SecGW's public key, hash of DPS's public key, nonce Nb, shared key Kab encrypted using DPS's public key, SecGW's public key, nonce Nseq, and attribute data_echo_Request.

Statement S3 states that SecGW receives from DPS in step three (at time t3) a message, which contains hash of DPS's public key, hash of SecGW's public key, function of nonce Nb, shared key Kab encrypted using SecGW's public key, DPS's public key, nonce Nseq, and attribute data_echo_Response.

- **Protocol Goals**

The major goal of HIP-based secure SDMN protocol is to protect the communication channels against IP based attacks such as DoS, reset, spoofing, replay and eavesdropping attacks.

These objectives are specified in the goals as follows:

```
G1: SecGW possess at[1] H(KaPub);
G2: SecGW possess at[1] H(KbPub);
G3: SecGW possess at[1] Nseq;
G4: DPS possess at[2] H(KbPub);
G5: DPS possess at[2] H(KaPub);
G6: DPS possess at[2] Kab;
G7: DPS possess at[2] data_echo_Request;
G8: DPS possess at[2] {Nb, {Kab}KaPub, KbPub, Nseq,
    data_echo_Request}KbPriv;
G9: DPS know at[2] SecGW send at[2] {Nb, {Kab}KaPub, KbPub, Nseq,
    data_echo_Request}KbPriv;
G10: DPS know at[2] (not(Zero possess at[0] H(KbPub), H(KaPub),
     {Nb, {Kab}KaPub, KbPub, Nseq, data_echo_Request}KbPriv));
G11: DPS know at[2] (not(Zero send at[0] H(KbPub), H(KaPub), {Nb,
     {Kab}KaPub, KbPub, Nseq, data_echo_Request}KbPriv));
G12: SecGW possess at[3] F(Nb);
G13: SecGW possess at[3] Kab;
G14: SecGW possess at[3] data_echo_Response;
G15: SecGW possess at[3] {F(Nb), {Kab}KbPub, KaPub, Nseq,
     data_echo_Response}KaPriv;
G16: SecGW know at[3] DPS send at[3] {F(Nb), {Kab}KbPub, KaPub,
     Nseq, data_echo_Response}KaPriv;
G17: SecGW know at[3] (not(Zero possess at[0] F(Nb)));
G18: SecGW know at[3] (not(Zero possess at[0] H(KaPub), H(KbPub),
     {F(Nb), {Kab}KbPub, KaPub, Nseq,data_echo_Response}KaPriv));
G19: SecGW know at[3] (not(Zero send at[0] H(KaPub), H(KbPub),
     {F(Nb), {Kab}KbPub, KaPub, Nseq,data_echo_Response}KaPriv));
```

Goal G1 states that SecGW possesses H(KaPub), G2 states that SecGW possesses H(KbPub) and goal G3 states that SecGW possesses Nseq after step 1 is completed. Goals G4, G5, G6, G7 and

G8 states that DPS possesses H(KbPub), H(KaPub), Kab, data_echo_Request, and {Nb, {Kab}KaPub, KbPub, Nseq, data_echo_Request}KbPriv respectively, after step 2 is completed.

Goal G9 states that DPS knows that the message {Nb, {Kab}KaPub, KbPub, Nseq, data_echo_Request}KbPriv was sent by SecGW, i.e, DPS knows that SecGW is certainly the source of this message. G10 states that DPS also knows that this message is fresh, i.e. it has been created by SecGW for the current protocol run. G11 states that no other entity has sent the same message to DPS. These steps represent authentication of SecGW to DPS.

Goals G12, G13, G14, and G15 states that SecGW possesses F(Nb), Kab, data_echo_Response, and {F(Nb), {Kab}KbPub, KaPub, Nseq, data_echo_Response}KaPriv respectively after step 3 is completed.

G16 states that SecGW knows that the message {F(Nb), {Kab}KbPub, KaPub, Nseq, data_echo_Response}KaPriv was sent by DPS, i.e, SecGW knows that DPS is certainly the source of this message. G17 states that F(Nb) is fresh and no other entity possesses F(Nb) at step 3. G18 states that SecGW also knows that this message is fresh, i.e. it has been created by DPS for the current protocol run. G19 states that no other entity has sent the same message to SecGW. These steps represent authentication of DPS to SecGW.

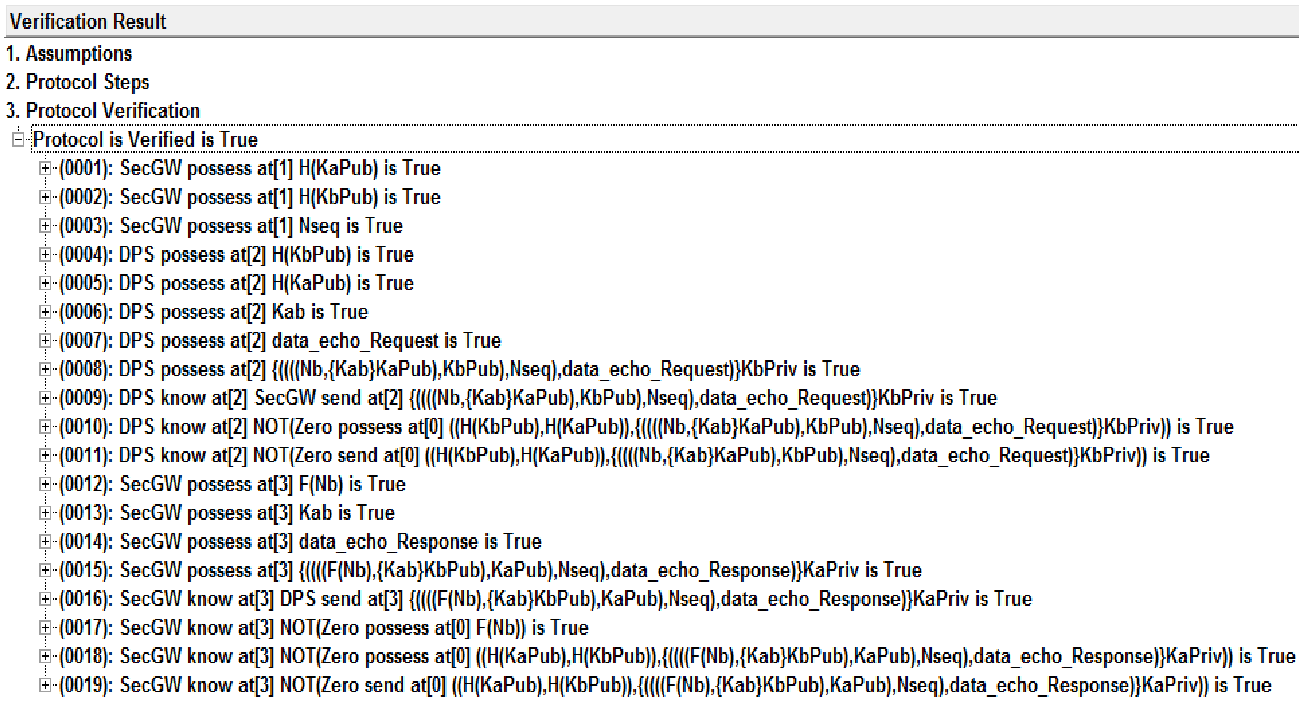
8.3.3 Verification results of HIP-based control channel in CDVT

The verification results of HIP-based control channel show that the protocol is true, i.e, it satisfies all the goals specified and the tool found no weaknesses in the protocol. This protocol [13] successfully performs authentication of DPSs and SecGWs, and establishes a secure HIP tunnel between them for communication.

The HIP-based control channel is safe against replay attacks as it is clear from goals G11 and G19 which states that no other entity other than the one it has received from, has sent the same message. Since these two goals are verified to be true in the CDVT tool, it corresponds to the fact that this protocol also resists man-in-the-middle (MITM) attack.

Goal G17, stating that F(Nb) is fresh and no other entity possesses F(Nb), is verified to be true in the CDVT tool. F(Nb) is a response to the challenge Nb (nonce sent by SecGW), and this goal being true corresponds to the freshness of the messages.

Goals G9, G10 and G11 are verified to be true and therefore, the authentication of SecGW to DPS is successful. Similarly, goals G16, G17, G18 and G19 are also verified to be true and the authentication of DPS to SecGW is successful.



```
Verification Result
1. Assumptions
2. Protocol Steps
3. Protocol Verification
⊟ Protocol is Verified is True
  ⊞ (0001): SecGW possess at[1] H(KaPub) is True
  ⊞ (0002): SecGW possess at[1] H(KbPub) is True
  ⊞ (0003): SecGW possess at[1] Nseq is True
  ⊞ (0004): DPS possess at[2] H(KbPub) is True
  ⊞ (0005): DPS possess at[2] H(KaPub) is True
  ⊞ (0006): DPS possess at[2] Kab is True
  ⊞ (0007): DPS possess at[2] data_echo_Request is True
  ⊞ (0008): DPS possess at[2] {((((Nb,{Kab}KaPub),KbPub),Nseq),data_echo_Request)}KbPriv is True
  ⊞ (0009): DPS know at[2] SecGW send at[2] {((((Nb,{Kab}KaPub),KbPub),Nseq),data_echo_Request)}KbPriv is True
  ⊞ (0010): DPS know at[2] NOT(Zero possess at[0] ((H(KbPub),H(KaPub)),{((((Nb,{Kab}KaPub),KbPub),Nseq),data_echo_Request)}KbPriv)) is True
  ⊞ (0011): DPS know at[2] NOT(Zero send at[0] ((H(KbPub),H(KaPub)),{((((Nb,{Kab}KaPub),KbPub),Nseq),data_echo_Request)}KbPriv)) is True
  ⊞ (0012): SecGW possess at[3] F(Nb) is True
  ⊞ (0013): SecGW possess at[3] Kab is True
  ⊞ (0014): SecGW possess at[3] data_echo_Response is True
  ⊞ (0015): SecGW possess at[3] {((((F(Nb),{Kab}KbPub),KaPub),Nseq),data_echo_Response)}KaPriv is True
  ⊞ (0016): SecGW know at[3] DPS send at[3] {((((F(Nb),{Kab}KbPub),KaPub),Nseq),data_echo_Response)}KaPriv is True
  ⊞ (0017): SecGW know at[3] NOT(Zero possess at[0] F(Nb)) is True
  ⊞ (0018): SecGW know at[3] NOT(Zero possess at[0] ((H(KaPub),H(KbPub)),{((((F(Nb),{Kab}KbPub),KaPub),Nseq),data_echo_Response)}KaPriv)) is True
  ⊞ (0019): SecGW know at[3] NOT(Zero send at[0] ((H(KaPub),H(KbPub)),{((((F(Nb),{Kab}KbPub),KaPub),Nseq),data_echo_Response)}KaPriv)) is True
```

Figure 14: Verification result of HIP-based control channel in CDVT

8.3.4 HIP-based data channel

The following are the notations used in the formal specification of the tunnel establishment procedure of data channel [13] in SDMN architecture:

- DPS1 : Data Plane Switch 1(A)
- DPS2 : Data Plane Switch 2(B)
- Nseq : Unique session identifier

- Nb : Nonce of DPS2(B)
- KaPub, KbPub : Public keys of DPS1(A) and DPS2(B) respectively
- KaPriv, KbPriv : Private keys of DPS1(A) and DPS2(B) respectively

The following are the steps of the protocol:

1. DPS1 -> DPS2 : H(KaPub), H(KbPub), Nseq
2. DPS2 -> DPS1 : H(KbPub), H(KaPub), {Nb, KbPub, Nseq, data_echo_Request}KbPriv
3. DPS1 -> DPS2 : H(KaPub), H(KbPub), {F(Nb), KaPub, Nseq, data_echo_Response}KaPriv

8.3.5 Formal Verification of HIP-based data channel

- **Initial Assumptions**

Initial assumptions are statements defining what each principal possesses and knows at the beginning of a protocol run.

The initial assumptions for HIP-based data channel of SDMN is as follows:

```
A1: DPS1 possess at[0] KaPub;
A2: DPS1 possess at[0] KaPriv;
A3: DPS1 possess at[0] KbPub;
A4: DPS1 possess at[0] Nseq;
A5: DPS1 know at[0] NOT(Zero possess at[0] Nseq);
A6: DPS1 know at[0] DPS2 possess at[0] KbPriv;
A7: DPS2 possess at[0] KbPub;
A8: DPS2 possess at[0] KbPriv;
A9: DPS2 possess at[0] KaPub;
A10: DPS2 possess at[0] Nb;
A11: DPS2 know at[0] NOT(Zero possess at[0] Nb);
A12: DPS2 know at[0] DPS1 possess at[0] KaPriv;
```

Assumptions A1, A2, A3 and A4 states that principal DPS1 possesses KaPub, KaPriv, KbPub and Nseq respectively. Assumption A5 represents the fact that before start of the protocol run, principal DPS1 is aware that no other principal possesses this nonce Nseq at that time. A6 states that DPS1 knows principal DPS2 possesses KbPriv before the start of the protocol run.

Similarly, assumptions A8, A9, A10 and A11 states that principal DPS2 possesses KbPub, KbPriv, KaPub and Nb respectively. Assumption A12 represents the fact that before start of the protocol run, principal DPS2 is aware that no other principal possesses this nonce Nb at that time. A13 states that DPS2 knows principal DPS1 possesses KaPriv before the start of the protocol run.

- **Protocol Steps**

The HIP-based secure SDMN data channel protocol steps are formalized as follows:

```
S1: DPS2 receivefrom DPS1 at[1] H(KaPub), H(KbPub), Nseq;
S2: DPS1 receivefrom DPS2 at[2] H(KbPub), H(KaPub), {Nb, DPS1,
    KbPub, Nseq, data_echo_Request}KbPriv;
S3: DPS2 receivefrom DPS1 at[3] H(KaPub), H(KbPub), {F(Nb), DPS2,
    KaPub, Nseq, data_echo_Response}KaPriv;
```

Statement S1 states that DPS2 receives from principal DPS1 in step one (at time t1) a message, which contains hash of DPS1's public key, hash of DPS2's public key and a nonce, Nseq.

S2 states that DPS1 receives from DPS2 in step two (at time t2) a message, which contains hash of DPS2's public key, hash of DPS1's public key, nonce Nb, identity of DPS1, DPS2's public key, nonce Nseq, and attribute data_echo_Request.

Statement S3 states that DPS2 receives from DPS1 in step three (at time t3) a message, which contains hash of DPS1's public key, hash of DPS2's public key, function of nonce Nb, identity of DPS2, DPS1's public key, nonce Nseq, and attribute data_echo_Response.

- **Protocol Goals**

The major goal of HIP-based secure SDMN protocol is to protect the communication channels against IP based attacks such as DoS, reset, spoofing, replay and eavesdropping attacks.

These objectives are specified in the goals as follows:

```
G1: DPS2 possess at[1] H(KaPub);
G2: DPS2 possess at[1] H(KbPub);
G3: DPS2 possess at[1] Nseq;
G4: DPS1 possess at[2] H(KbPub);
G5: DPS1 possess at[2] H(KaPub);
G6: DPS1 possess at[2] Nb;
G7: DPS1 possess at[2] Nseq;
G8: DPS1 possess at[2] data_echo_Request;
G9: DPS1 possess at[2] {Nb, DPS1, KbPub, Nseq,
     data_echo_Request}KbPriv;
G10: DPS1 know at[2] DPS2 send at[2] {Nb, DPS1, KbPub, Nseq,
     data_echo_Request}KbPriv;
G11: DPS1 know at[2] (not(Zero possess at[0] H(KbPub), H(KaPub),
     {Nb, DPS1, KbPub, Nseq, data_echo_Request}KbPriv));
G12: DPS1 know at[2] (not(Zero send at[0] H(KbPub), H(KaPub),
     {Nb, DPS1, KbPub, Nseq, data_echo_Request}KbPriv));
G13: DPS2 possess at[3] F(Nb);
G14: DPS2 possess at[3] data_echo_Response;
G15: DPS2 possess at[3] {F(Nb), DPS2, KaPub, Nseq,
     data_echo_Response}KaPriv;
G16: DPS2 know at[3] DPS1 send at[3] {F(Nb), DPS2, KaPub, Nseq,
     data_echo_Response}KaPriv;
G17: DPS2 know at[3] (not(Zero possess at[0] F(Nb)));
G18: DPS2 know at[3] (not(Zero possess at[0] H(KaPub), H(KbPub),
     {F(Nb), DPS2, KaPub, Nseq, data_echo_Response}KaPriv));
```

```
G19: DPS2 know at[3] (not(Zero send at[0] H(KaPub), H(KbPub),
     {F(Nb), DPS2, KaPub, Nseq, data_echo_Response}KaPriv));
```

Goal G1, G2 and G3 states that DPS2 possesses H(KaPub), H(KbPub) and Nseq, respectively after step 1 is completed. Goals G4, G5, G6, G7 and G8 states that DPS1 possesses H(KbPub), H(KaPub), Nb, Nseq and data_echo_Request respectively, after step 2 is completed.

Goal G9 states that DPS1 possesses {Nb, DPS1, KbPub, Nseq, data_echo_Request}KbPriv after step 2 is completed. G10 states that DPS1 knows that the message {Nb, DPS1, KbPub, Nseq, data_echo_Request}KbPriv was sent by DPS2, i.e, DPS1 knows that DPS2 is certainly the source of this message. G11 states that DPS1 also knows that this message is fresh, i.e. it has been created by DPS2 for the current protocol run. G12 states that no other entity has sent the same message to DPS1. These steps represent authentication of DPS2 to DPS1.

Goals G13, G14 and G15 states that DPS2 possesses F(Nb), data_echo_Response, and {F(Nb), DPS2, KaPub, Nseq, data_echo_Response}KaPriv respectively after step 3 is completed. G16 states that DPS2 knows that the message {F(Nb), DPS2, KaPub, Nseq, data_echo_Response}KaPriv was sent by DPS1, i.e, DPS2 knows that DPS1 is certainly the source of this message.

G17 states that F(Nb) is fresh and no other entity possesses F(Nb) at step 3. G18 states that DPS2 also knows that the message {F(Nb), DPS2, KaPub, Nseq, data_echo_Response}KaPriv is fresh, i.e. it has been created by DPS1 for the current protocol run. G19 states that no other entity has sent the same message to DPS2. These steps represent authentication of DPS1 to DPS1.

8.3.6 Verification results of HIP-based data channel in CDVT

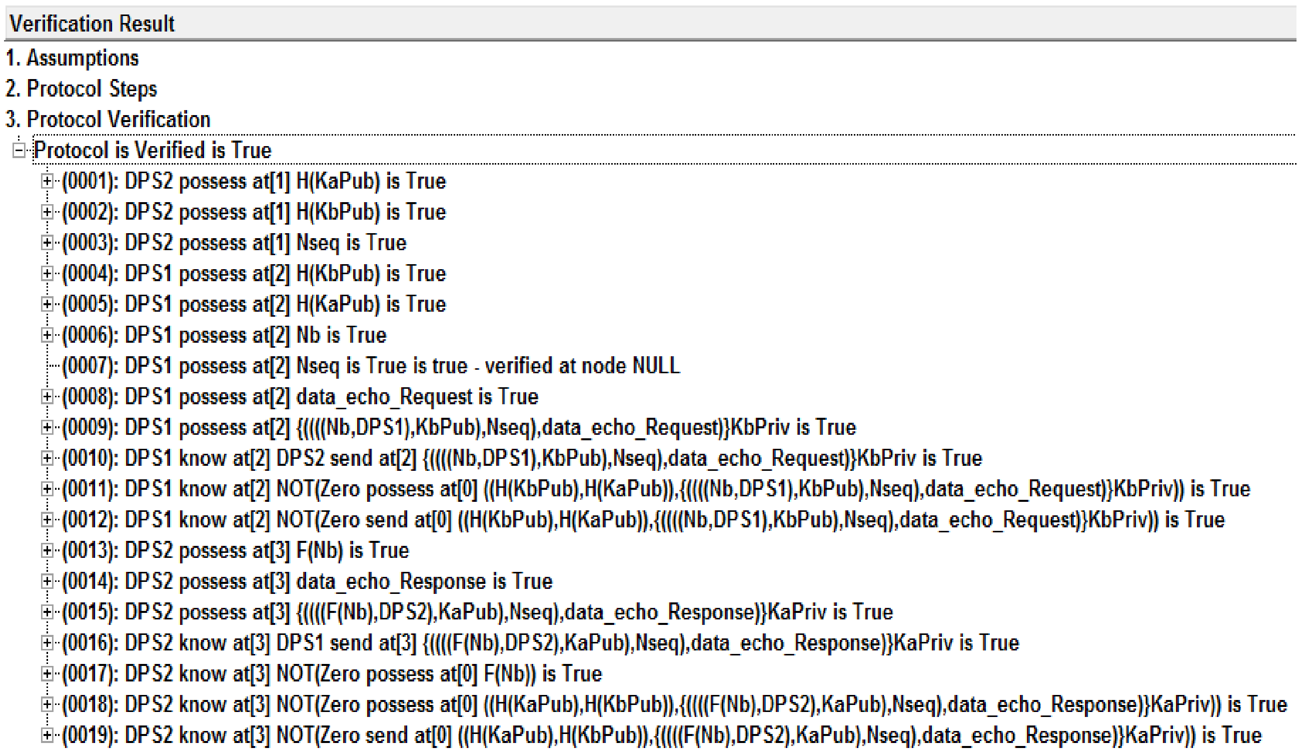
The verification results of HIP-based data channel show that the protocol is true, i.e, it satisfies all the goals specified and the tool found no weaknesses in the protocol.

This protocol [13] successfully performs mutual authentication between DPSs, and establishes a secure IPSec tunnel between them for communication.

The HIP-based data channel is safe against replay attacks as it is clear from goals G12 and G19 which states that no other entity (possible intruders), has sent the same message. Since these two goals are verified to be true in the CDVT tool, it corresponds to the fact that this protocol also resists man-in-the-middle (MITM) attack.

Goal G17, stating that F(Nb) is fresh and no other entity possesses F(Nb), is verified to be true in the CDVT tool. F(Nb) is a response to the challenge Nb (nonce sent by DPS2), and this goal being true corresponds to the freshness of the messages.

Goals G10, G11 and G12 are verified to be true and therefore, the authentication of DPS2 to DPS1 is successful. Similarly, goals G16, G17, G18 and G19 are also verified to be true and the authentication of DPS1 to DPS2 is successful.



```
Verification Result
1. Assumptions
2. Protocol Steps
3. Protocol Verification
⊟ Protocol is Verified is True
   ⊞ (0001): DPS2 possess at[1] H(KaPub) is True
   ⊞ (0002): DPS2 possess at[1] H(KbPub) is True
   ⊞ (0003): DPS2 possess at[1] Nseq is True
   ⊞ (0004): DPS1 possess at[2] H(KbPub) is True
   ⊞ (0005): DPS1 possess at[2] H(KaPub) is True
   ⊞ (0006): DPS1 possess at[2] Nb is True
     (0007): DPS1 possess at[2] Nseq is True is true - verified at node NULL
   ⊞ (0008): DPS1 possess at[2] data_echo_Request is True
   ⊞ (0009): DPS1 possess at[2] {((((Nb,DPS1),KbPub),Nseq),data_echo_Request)}KbPriv is True
   ⊞ (0010): DPS1 know at[2] DPS2 send at[2] {((((Nb,DPS1),KbPub),Nseq),data_echo_Request)}KbPriv is True
   ⊞ (0011): DPS1 know at[2] NOT(Zero possess at[0] ((H(KbPub),H(KaPub)),{((((Nb,DPS1),KbPub),Nseq),data_echo_Request)}KbPriv)) is True
   ⊞ (0012): DPS1 know at[2] NOT(Zero send at[0] ((H(KbPub),H(KaPub)),{((((Nb,DPS1),KbPub),Nseq),data_echo_Request)}KbPriv)) is True
   ⊞ (0013): DPS2 possess at[3] F(Nb) is True
   ⊞ (0014): DPS2 possess at[3] data_echo_Response is True
   ⊞ (0015): DPS2 possess at[3] {((((F(Nb),DPS2),KaPub),Nseq),data_echo_Response)}KaPriv is True
   ⊞ (0016): DPS2 know at[3] DPS1 send at[3] {((((F(Nb),DPS2),KaPub),Nseq),data_echo_Response)}KaPriv is True
   ⊞ (0017): DPS2 know at[3] NOT(Zero possess at[0] F(Nb)) is True
   ⊞ (0018): DPS2 know at[3] NOT(Zero possess at[0] ((H(KaPub),H(KbPub)),{((((F(Nb),DPS2),KaPub),Nseq),data_echo_Response)}KaPriv)) is True
   ⊞ (0019): DPS2 know at[3] NOT(Zero send at[0] ((H(KaPub),H(KbPub)),{((((F(Nb),DPS2),KaPub),Nseq),data_echo_Response)}KaPriv)) is True
```

Figure 15: Verification result of HIP-based data channel in CDVT

8.4 Secure Mobile Wallet

The secure mobile wallet protocol [14] has been developed to increase the security of mobile wallets and customers by introducing user anonymity or privacy preserving feature during mobile payment. These features are obtained using digital signatures and pseudo-identity techniques. In order to ease the computation overhead in mobile devices with limited resources, this protocol uses cloud computing where the computation task during signature verification is outsourced to an untrusted cloud server in a secure way.

The notations used in the formal specification of secure mobile wallet [14] is as follows:

- Alice : Customer
- Bob : Merchant
- TTP : Trusted third party
- tId : Transaction identity
- idA : Real identity of Alice
- idB : Real identity of Bob
- psuidA : Psuedo-identity of Alice
- psuidB : Psuedo-identity of Bob
- amountT : Amount to be paid
- amountR : Amount received
- KaPub, KaPriv : Public and private keys of Alice
- KbPub, KbPriv : Public and private keys of Bob
- dA, dB : Partial private keys of Alice and Bob respectively
- KttpPub, KttpPriv : Public and private keys of TTP

The following are the steps of the protocol:

1. Alice -> TTP : {idA}KttpPub
2. TTP -> Alice : {psuidA, dA}KttpPriv
3. Bob -> TTP : {idB}KttpPub
4. TTP -> Bob : {psuidB, dB}KttpPriv

5. Bob -> Alice : payment = (tId, amountT, psuidB)

   Alice : Signature = {dA, H(payment, psuidA, KaPub)}KaPriv
6. Alice -> Bob : (Signature, payment, KaPub)

   Bob : Signature = {dB, H(tId, amountR, psuidB, KaPub)}KbPriv
7. Bob -> Alice : (tId, amountR, Signature)

8.4.1 Formal Verification of Secure Mobile Wallet

- **Initial Assumptions**

Initial assumptions are statements defining what each principal possesses and knows at the beginning of a protocol run.

The initial assumptions for secure mobile wallet protocol is as follows:

```
A1: A possess at[0] KaPub;
A2: A possess at[0] KaPriv;
A3: A possess at[0] idA;
A4: A possess at[0] KbPub;
A5: A possess at[0] KttpPub;
A6: A know at[0] TTP possess at[0] KttpPriv;
A7: A know at[0] B possess at[0] KbPriv;
A8: A know at[0] NOT(Zero possess at[0] KaPriv);
A9: A know at[0] NOT(Zero possess at[0] idA);
A10: A know at[0] NOT(Zero possess at[0] psuidA);
A11: A know at[0] NOT(Zero possess at[0] dA);
A12: A know at[0] NOT(Zero possess at[0] dB);
A13: A know at[0] NOT(Zero possess at[0] tId);
A14: B possess at[0] KbPub;
A15: B possess at[0] KbPriv;
A16: B possess at[0] idB;
A17: B possess at[0] amountT;
A18: B possess at[0] amountR;
```

```
A19: B possess at[0] tId;
A20: B possess at[0] KaPub;
A21: B possess at[0] KttpPub;
A22: B know at[0] TTP possess at[0] KttpPriv;
A23: B know at[0] A possess at[0] KaPriv;
A24: B know at[0] NOT(Zero possess at[0] KbPriv);
A25: B know at[0] NOT(Zero possess at[0] idB);
A26: B know at[0] NOT(Zero possess at[0] tId);
A27: B know at[0] NOT(Zero possess at[0] psuidB);
A28: TTP possess at[0] KttpPub;
A29: TTP possess at[0] KttpPriv;
A30: TTP possess at[0] psuidA;
A31: TTP possess at[0] psuidB;
A32: TTP possess at[0] dA;
A33: TTP possess at[0] dB;
A34: TTP possess at[0] KaPub;
A35: TTP possess at[0] KbPub;
A36: TTP know at[0] A possess at[0] KaPriv;
A37: TTP know at[0] B possess at[0] KbPriv;
A38: TTP know at[0] NOT(Zero possess at[0] KttpPriv);
A39: TTP know at[0] NOT(Zero possess at[0] psuidA);
A40: TTP know at[0] NOT(Zero possess at[0] psuidB);
A41: TTP know at[0] NOT(Zero possess at[0] dA);
A42: TTP know at[0] NOT(Zero possess at[0] dB);
A43: TTP know at[0] NOT(Zero possess at[0] idA);
A44: TTP know at[0] NOT(Zero possess at[0] idB);
```

Assumptions A1, A2, A3, A4 and A5 states that principal A possesses KaPub, KaPriv, idA, KbPub and KttpPub respectively. Assumptions A6 and A7 states that A knows TTP possesses KttpPriv and B possesses KbPriv before the start of the protocol run. Assumptions A8 to A13 represents the fact that before start of the protocol run, principal A is aware that no other principal possesses KaPriv, idA, pseudo-identity psuidA, partial keys dA and dB, and transaction ID tId at that time.

Similarly, assumptions A14, A5, A16, A17, A18, A19, A20 and A21 states that principal B possesses KbPub, KbPriv, idB, amountT, amountR, tId, KaPub and KttpPub respectively. A22 and A23 states that B knows TTP possesses KttpPriv and A possesses KaPriv before the start of the protocol run. A24 to A27 represents the fact that before start of the protocol run, principal B is aware that no other principal possesses KbPriv, idB, tId and psuidB at that time.

A28 to A35 states that TTP possesses before the start of the protocol run, KttpPub, KttpPriv, psuidA, psuidB, dA, dB, KaPub and KbPub. A36 and A37 states that TTP knows A possesses KaPriv and B possesses KbPriv before the start of the protocol run. A38 to A44 represents the fact that before start of the protocol run, principal TTP is aware that no other principal possesses KttpPriv, psuidA, psuidB, dA, dB, idA and idB at that time.

- **Protocol Steps**

The secure mobile wallet protocol steps are formalized as follows:

```
S1: TTP receivefrom A at[1] {{idA}KttpPub}KaPriv;
S2: A receivefrom TTP at[2] {{psuidA, dA}KaPub}KttpPriv;
S3: TTP receivefrom B at[3] {{idB}KttpPub}KbPriv;
S4: B receivefrom TTP at[4] {{psuidB, dB}KbPub}KttpPriv;
S5: A receivefrom B at[5] {{tId, amountT, psuidB}KaPub}KbPriv;
S6: B receivefrom A at[6] {{dA, H(tId, amountT, psuidB, psuidA,
    KaPub)}KaPriv, tId, amountT, psuidB, KaPub}KbPub;
S7: A receivefrom B at[7] {{tId, amountR, {dB, H(tId, amountR,
    psuidB, KaPub)}KbPriv}KaPub}KbPriv;
```

Statement S1 states that TTP receives from principal A in step one (at time t1) a message, which contains the real identity of A, idA encrypted using TTP's public key. S2 states that A receives from TTP in step two (at time t2) a message, which contains the pseudo-identity of A, psuidA, and partial private key of A, dA, both encrypted using A's public key.

Statement S3 states that TTP receives from principal B in step three (at time t3) a message, which contains the real identity of B, idB encrypted using TTP's public key. S4 states that B receives from TTP in step four (at time t4) a message, which contains the pseudo-identity of B, psuidB, and partial private key of B, dB, both encrypted using B's public key.

S5 states that A receives from B at time t5 the transaction identity tId, the amount to be transmitted, amountT, pseudo-identity of B psuidB, all encrypted using A's public key. S6 states that B receives from A at time t6 the partial private key dA, hash of tId, amountT, psuidB, psuidA and A's public key, encrypted using B's public key. S7 states that A receives from B at time t7 tId, amount received from A amountR, partial private key of B dB, hash of tId, amountR, psuidB, KaPub, all encrypted using A's public key.

- **Protocol Goals**

The major goal of secure mobile wallet is to preserve the privacy of the user, restrict the traceability, and small overhead.

These objectives are specified in the goals as follows:

```
G1: TTP possess at[1] idA;
G2: TTP know at[1] A send at[1] {{idA}KttpPub}KaPriv;
G3: TTP know at[1] NOT(Zero possess at[0] {{idA}KttpPub}KaPriv);
G4: A possess at[2] psuidA;
G5: A possess at[2] dA;
G6: A know at[2] TTP send at[2] {{psuidA, dA}KaPub}KttpPriv;
G7: A know at[2] NOT(Zero possess at[0] {{psuidA, dA} KaPub}
     KttpPriv);
G8: TTP possess at[3] idB;
G9: TTP know at[3] B send at[3] {{idB}KttpPub}KbPriv;
G10: TTP know at[3] NOT(Zero possess at[0] {{idB}KttpPub}KbPriv);
G11: B possess at[4] psuidB;
G12: B possess at[4] dB;
G13: B know at[4] TTP send at[4] {{psuidB, dB}KbPub}KttpPriv;
```

```
G14: B know at[4] NOT(Zero possess at[0] {{psuidB, dB} KbPub}
     KttpPriv);
G15: A possess at[5] tId;
G16: A possess at[5] amountT;
G17: A possess at[5] psuidB;
G18: A know at[5] B send at[5] {{tId, amountT, psuidB} KaPub}
     KbPriv;
G19: A know at[5] NOT(Zero possess at[0] {{tId, amountT, psuidB}
     KaPub} KbPriv);
G20: B possess at[6] {dA, H(tId, amountT, psuidB, psuidA, KaPub)}
     KaPriv;
G21: B know at[6] A send at[6] {dA, H(tId, amountT, psuidB,
     psuidA, KaPub)}KaPriv;
G22: B know at[6] NOT(Zero possess at[0] {dA, H(tId, amountT,
     psuidB, psuidA, KaPub)}KaPriv);
G23: A possess at[7] {dB, H(tId, amountR, psuidB, KaPub)}KbPriv;
G24: A possess at[7] amountR;
G25: A know at[7] B send at[7] {{tId, amountR, {dB, H(tId,
     amountR, psuidB, KaPub)}KbPriv}KaPub}KbPriv;
G26: A know at[7] NOT(Zero possess at[0] {{tId, amountR, {dB,
     H(tId, amountR, psuidB, KaPub)}KbPriv}KaPub}KbPriv);
```

Goal G1 states that TTP possesses idA after step 1 is completed. Goal G2 states that TTP knows A is the source of the message {idA}KttpPub. G3 states that TTP also knows that this message is fresh, i.e. it has been created by A for the current protocol run. Goals G4 and G5 states that A possesses psuidA and dA after step 2 is completed. Goal G6 states that A knows TTP is the source of the message {psuidA, dA}KaPub. G7 states that A also knows that this message is fresh, i.e. it has been created by TTP for the current protocol run.

Goal G8 states that TTP possesses idB after step 3 is completed. Goal G9 states that TTP knows B is the source of the message {idB}KttpPub. G10 states that TTP also knows that this message is fresh, i.e. it has been created by B for the current protocol run. Goals G11 and G12 states that B possesses psuidB and dB after step 4 is completed. Goal G13 states that B knows TTP is the source of the message {psuidB, dB}KbPub. G14 states that B also knows that this message is fresh, i.e. it has been created by TTP for the current protocol run.

Goals G15, G16 and G17 states that A possesses tId, amountT and psuidB respectively after step 5 is completed. G18 states that A knows that the message {tId, amountT, psuidB}KaPub was sent by B, i.e, A knows that B is certainly the source of this message. G19 states that A also knows that this message is fresh, i.e. it has been created by B for the current protocol run.

Goals G20 states that B possesses dA, H(tId, amountT, psuidB, psuidA, KaPub) after step 6 is completed. G21 states that B knows that the message dA, H(tId, amountT, psuidB, psuidA, KaPub) was sent by A, i.e, B knows that A is certainly the source of this message. G22 states that B also knows that this message is fresh, i.e. it has been created by A for the current protocol run.

G23 states that A possess dB, H(tId, amountR, psuidB, KaPub) after step 7 is completed. G24 states A possesses the amount received, amountR. G25 states that A knows that the message dB, H(tId, amountR, psuidB, KaPub) was sent by B, i.e, A knows that B is certainly the source of this message. G26 states that A also knows that this message is fresh, i.e. it has been created by B for the current protocol run.

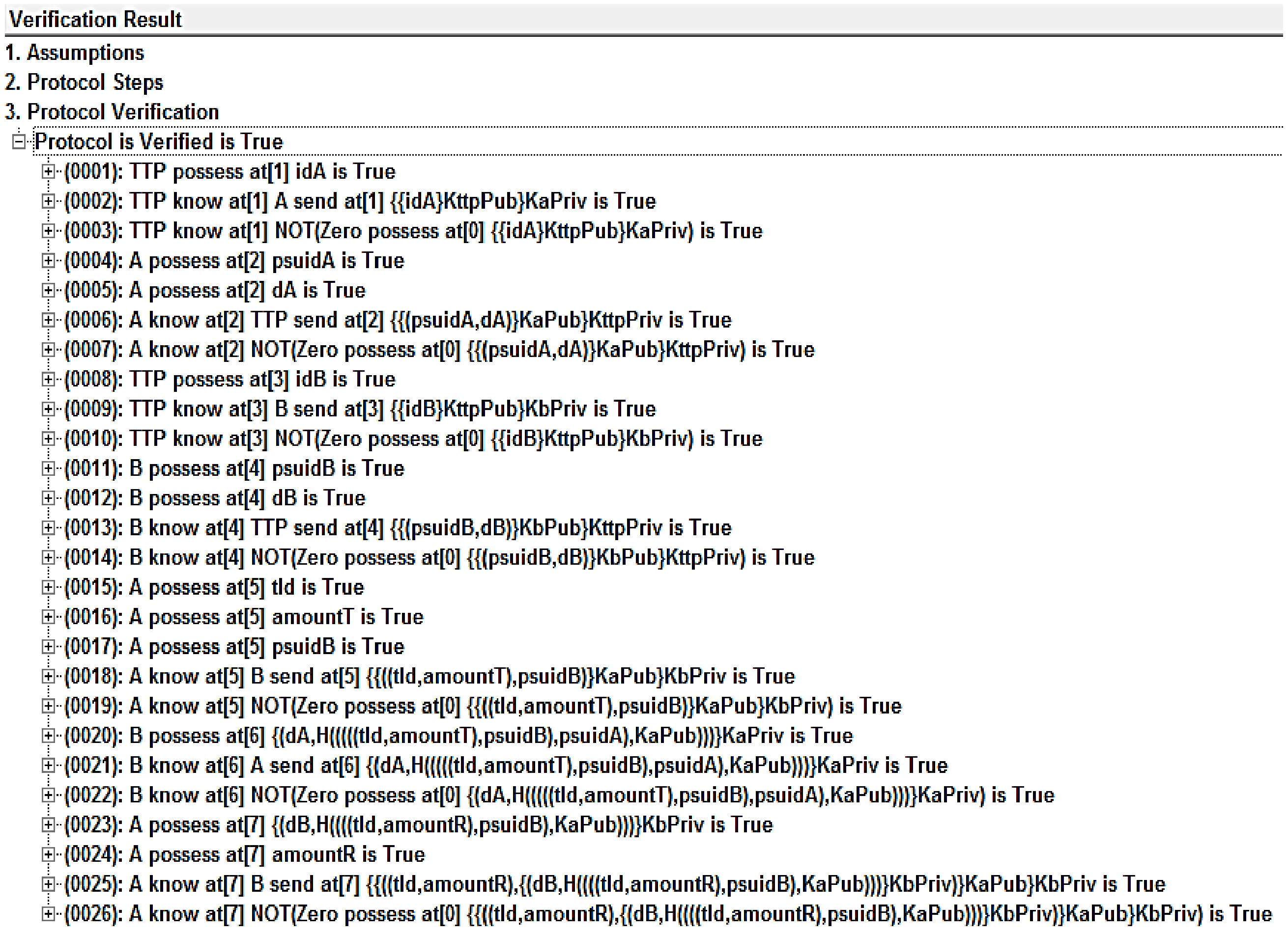
8.4.2 Verification results of secure mobile wallet in CDVT

The verification results of secure mobile wallet show that the protocol is true, i.e, it satisfies all the goals specified and the tool found no weaknesses in the protocol. This protocol [14] is successful in obtaining security of data during payment and also provides user anonymity. However, the verification of digital signature cannot be fully represented in the CDVT tool, and hence we assume here that the verifications are successful and that the users are genuine, as the protocol has a very strong digital signature generation and verification procedure which when analyzed proved to be secure.

The goals G3, G7, G10 and G14 are verified to be true, which in turn proves that the freshness of messages is maintained. This is because the method of generating public and secret keys is secure and contains no weaknesses.

Also, user anonymity is maintained with the use of pseudo-identities instead of real identities during the payment phase, as only the TTP (PSP, here) knows the real identities of the entities. Only the TTP can trace back to the real identity of an entity as it created the pseudo-identities, proved to be true in goals G6 and G13. Since the TTP can trace back to real identities, it has the potential to find malicious users.

This protocol also satisfies non-repudiation properties as a merchant cannot repudiate the origin of the receipt and the customer cannot repudiate the origin of the payment. This has been implemented by of the use of digital signature and the partial private keys dA and dB in them, as mentioned in goals G20, G22, G23 and G26.



```
Verification Result
1. Assumptions
2. Protocol Steps
3. Protocol Verification
⊟ Protocol is Verified is True
    ⊞ (0001): TTP possess at[1] idA is True
    ⊞ (0002): TTP know at[1] A send at[1] {{idA}KttpPub}KaPriv is True
    ⊞ (0003): TTP know at[1] NOT(Zero possess at[0] {{idA}KttpPub}KaPriv) is True
    ⊞ (0004): A possess at[2] psuidA is True
    ⊞ (0005): A possess at[2] dA is True
    ⊞ (0006): A know at[2] TTP send at[2] {{(psuidA,dA)}KaPub}KttpPriv is True
    ⊞ (0007): A know at[2] NOT(Zero possess at[0] {{(psuidA,dA)}KaPub}KttpPriv) is True
    ⊞ (0008): TTP possess at[3] idB is True
    ⊞ (0009): TTP know at[3] B send at[3] {{idB}KttpPub}KbPriv is True
    ⊞ (0010): TTP know at[3] NOT(Zero possess at[0] {{idB}KttpPub}KbPriv) is True
    ⊞ (0011): B possess at[4] psuidB is True
    ⊞ (0012): B possess at[4] dB is True
    ⊞ (0013): B know at[4] TTP send at[4] {{(psuidB,dB)}KbPub}KttpPriv is True
    ⊞ (0014): B know at[4] NOT(Zero possess at[0] {{(psuidB,dB)}KbPub}KttpPriv) is True
    ⊞ (0015): A possess at[5] tId is True
    ⊞ (0016): A possess at[5] amountT is True
    ⊞ (0017): A possess at[5] psuidB is True
    ⊞ (0018): A know at[5] B send at[5] {{((tId,amountT),psuidB)}KaPub}KbPriv is True
    ⊞ (0019): A know at[5] NOT(Zero possess at[0] {{((tId,amountT),psuidB)}KaPub}KbPriv) is True
    ⊞ (0020): B possess at[6] {(dA,H(((((tId,amountT),psuidB),psuidA),KaPub))))}KaPriv is True
    ⊞ (0021): B know at[6] A send at[6] {(dA,H(((((tId,amountT),psuidB),psuidA),KaPub))))}KaPriv is True
    ⊞ (0022): B know at[6] NOT(Zero possess at[0] {(dA,H(((((tId,amountT),psuidB),psuidA),KaPub))))}KaPriv) is True
    ⊞ (0023): A possess at[7] {(dB,H((((tId,amountR),psuidB),KaPub))))}KbPriv is True
    ⊞ (0024): A possess at[7] amountR is True
    ⊞ (0025): A know at[7] B send at[7] {{((tId,amountR),{(dB,H((((tId,amountR),psuidB),KaPub))))}KbPriv)}KaPub}KbPriv is True
    ⊞ (0026): A know at[7] NOT(Zero possess at[0] {{((tId,amountR),{(dB,H((((tId,amountR),psuidB),KaPub))))}KbPriv)}KaPub}KbPriv) is True
```

Figure 16: Verification Result of Secure Mobile Wallet in CDVT

8.5 Secure patient-health monitoring Wireless Body Area Networks

The secure patient-health monitoring WBAN protocol [15] mainly consists of two phases:

1. Communication Establishment Phase.

   This phase includes secure transfer of session key between sensor and doctor through the data sink, for future communications. The sensor generates a session key and sends it to the data sink in an encrypted form. The doctor obtains this key from data sink and decrypts using its private key and verifies itself to the sensor. This is called session key authentication.

2. Communication Phase

   In this phase, the session key obtained from previous phase is used to send instructions and messages between doctor and sensor.

The notations used in the formal specification of secure WBAN protocol [15] is as follows:

- S : Sensor
- D : Doctor
- DS : Data Sink
- Ksd : Random access token chosen by sensor
- Ksd1 : Shared secret between doctor and sensor
- KdPub : Public parameter generated by KGC to doctor
- KdPriv : Secret key distributed by KGC to doctor
- KsPub : Public parameter generated by KGC to sensor
- KsPriv : Secret key distributed by KGC to sensor
- TSs : Timestamp
- idD : ID of doctor
- idS : ID of sensor
- instruc : Instructions sent by doctor to sensor
- message : Data sent by sensor to doctor

The steps involved in the protocol are:

1. Sensor -> Data Sink : idS, {{Ktdate}KdPub, H(Ktdate)}KsPriv

   Sensor : Ktdate = Ksd||TSs

   Sensor : H = Hash(Ktdate)
2. Data Sink -> Doctor : idS, {{Ktdate}KdPub}KsPriv

   Doctor : {{{Ktdate}KdPub}KsPriv}KsPub = {{Ktdate}KdPub}KdPriv = Ktdate
3. Doctor -> Sensor : idD, H(Ktdate)

   Sensor : If H1 = H(Ktdate), Ktdate1 = Ksd1||TSs

   Sensor : H2 = H(Ktdate1)
4. Sensor -> Data Sink : idS, {{Ktdate1}KdPub, H(Ktdate1)}KsPriv
5. Sensor -> Doctor : idS, {{Ktdate1}KdPub}KsPriv

   Doctor : {{{Ktdate1}KdPub}KsPriv}KsPub = {{Ktdate1}KdPub}KdPriv = Ktdate1
6. Doctor -> Sensor : idD, H(Ktdate1)

   Sensor : If H2 = H(Ktdate1), continue
7. Doctor -> Sensor : {idD, idS, {instruc}Ksd1, H(Ksd1, instruc, idS)}KdPriv

   Sensor : {{{instruc}Ksd1}KdPriv}KdPub = {{instruc}Ksd1}Ksd1 = instruc

   Sensor : Computes H'(Ksd1, instruc, idS).

   If equal to H(Ksd1, instruc, idS), message integrity is proved.
8. Sensor -> Doctor : {idS, idD, {message}Ksd1}KsPriv

8.5.1 Formal Verification of patient-health monitoring protocol in WBAN

- **Initial Assumptions**

Initial assumptions are statements defining what each principal possesses and knows at the beginning of a protocol run.

The initial assumptions for patient-health monitoring WBAN protocol [15] is as follows:

```
A1: S possess at[0] Ksd;
A2: S possess at[0] Ksd1;
A3: S possess at[0] idS;
```

```
A4: S possess at[0] KsPub;
A5: S possess at[0] KsPriv;
A6: S possess at[0] message;
A7: S possess at[0] KdPub;
A8: S possess at[0] TSs;
A9: S know at[0] D possess at[0] KsPub;
A10: S know at[0] D possess at[0] idD;
A11: S know at[0] D possess at[0] KdPriv;
A12: S know at[0] not(Zero possess at[0] KsPriv);
A13: S know at[0] not(Zero possess at[0] Ksd);
A14: S know at[0] not(Zero possess at[0] Ksd1);
A15: S know at[0] not(Zero possess at[0] TSs);
A16: S know at[0] not(Zero possess at[0] idS);
A17: D possess at[0] idD;
A18: D possess at[0] KdPub;
A19: D possess at[0] KdPriv;
A20: D possess at[0] instruc;
A21: D possess at[0] KsPub;
A22: D know at[0] S possess at[0] KdPub;
A23: D know at[0] S possess at[0] idS;
A24: D know at[0] S possess at[0] Ksd;
A25: D know at[0] S possess at[0] Ksd1;
A26: D know at[0] S possess at[0] KsPriv;
A27: D know at[0] not(Zero possess at[0] KdPriv);
A28: D know at[0] not(Zero possess at[0] idD);
A29: D know at[0] not(Zero possess at[0] Ksd1);
A30: DS possess at[0] KdPub;
A31: DS possess at[0] KsPub;
A32: DS know at[0] D possess at[0] KdPriv;
A33: DS know at[0] S possess at[0] KsPriv;
A34: DS know at[0] not(Zero possess at[0] TSs);
A35: DS know at[0] not(Zero possess at[0] Ksd);
```

```
A36: DS know at[0] not(Zero possess at[0] Ksd1);
```

Assumptions A1, A2, A3, A4, A5, A6, A7 and A8 states that principal S possesses Ksd, Ksd1, idS, KsPub, KsPriv, message, KdPub, and TSs respectively. Assumption A9, A10 and A11 states that S knows principal D possesses KsPub, idD and KdPriv before the start of the protocol run. A12 to A16 represents the fact that before start of the protocol run, principal S is aware that no other principal possesses KsPriv, Ksd, Ksd1, TSs and idS at that time.

Similarly, assumptions A17, A18, A19, A20 and A21 represents the fact that principal D possesses idD, KdPub, KdPriv, instruc and KsPub respectively. A22, A23, A24, A25 and A26 expresses that D knows S possesses KdPub, idS, Ksd, Ksd1 and KsPriv at the same time. Assumptions A27, A28, A29 represents the fact that before start of the protocol run, principal D is aware that no other principal possesses KdPriv, idD and Ksd1 at that time.

A30 and A31 states that DS possesses KdPub and KsPub respectively. A32 and A33 expresses that DS knows D possess KdPriv and S possess KsPriv respectively. Assumptions A34, A35, A36 states that before start of the protocol run, principal DS is aware that no other principal possesses TSs, Ksd and Ksd1 at that time.

- **Protocol Steps**

The secure patient-health monitoring WBA protocol steps are formalized as follows:

```
S1: DS receivefrom S at[1] {idS,{Ksd,TSs}KdPub,H(Ksd,TSs)}KsPriv;
S2: D receivefrom DS at[2] idS, {Ksd,TSs}KdPub;
S3: S receivefrom D at[3] {idD, H(Ksd,TSs)}KdPriv;
S4: DS receivefrom S at[4] {idS,{Ksd1,TSs}KdPub,H(Ksd1,TSs)}
    KsPriv;
S5: D receivefrom S at[5] {idS, {Ksd1,TSs}KdPub}KsPriv;
S6: S receivefrom D at[6] {idD, H(Ksd1,TSs)}KdPriv;
S7: S receivefrom D at[7] {idD, idS, {instruc}Ksd1, H(Ksd1,
    instruc, idS)}KdPriv;
```

```
S8: D receivefrom S at[8] {idS, idD, {message}Ksd1}KsPriv;
```

Statement S1 states that DS receives from principal S in step one (at time t1) a message, which contains the identity of S, idS, access token Ksd with timestamp encrypted using D’s public key, and hash of Ksd with timestamp.

S2 states that D receives from DS in step two (at time t2) a message, which contains the identity of S, idS and access token Ksd with timestamp encrypted using D’s public key. Statement S3 states that S receives from D in step three (at time t3) a message, which contains the identity of D, idD and hash of Ksd with timestamp.

S4 states that DS receives from S in step four (at time t4) a message, which contains the identity of S, idS, new access token Ksd1 with timestamp encrypted using D’s public key, and hash of Ksd1 with timestamp. S5 states that D receives from S at time t5 the the identity of S, idS and new access token Ksd1 with timestamp encrypted using D’s public key. S6 states that S receives from D at time t6 the the identity of D, idD and hash of Ksd1 with timestamp.

S7 states that S receives from D at time t7 idD, idS, instructions encrypted using shared key Ksd1, and hash of Ksd1, instructions and idS. S8 states that D receives from S at time t8 idS, idD and message encrypted using shared key Ksd1..

- **Protocol Goals**

The major goal of secure patient-health monitoring WBAN protocol is to secure the data communication between sensors, data sink and data consumers by employing two-phase commitment and session key authentication.

These objectives are specified in the goals as follows:

```
G1: DS possess at[1] idS;
G2: DS possess at[1] {Ksd,TSs}KdPub;
G3: DS possess at[1] H(Ksd,TSs);
```

```
G4: DS know at[1] S send at[1] {idS, {Ksd,TSs}KdPub, H(Ksd,TSs)}
     KsPriv;
G5: DS know at[1] not(Zero possess at[0] {idS, {Ksd,TSs}KdPub,
     H(Ksd,TSs)}KsPriv);
G6: D possess at[2] idS;
G7: D possess at[2] Ksd;
G8: D possess at[2] TSs;
G9: D know at[2] S possess at[2] Ksd;
G10: D know at[2] DS send at[2] idS, {Ksd,TSs}KdPub;
G11: D know at[2] not(Zero possess at[0] idS, {Ksd,TSs}KdPub);
G12: S possess at[3] idD;
G13: S possess at[3] H(Ksd,TSs);
G14: S know at[3] D send at[3] {idD, H(Ksd,TSs)}KdPriv;
G15: S know at[3] not(Zero possess at[0] {idD,H(Ksd,TSs)} KdPriv);
G16: DS possess at[4] idS;
G17: DS possess at[4] {Ksd1,TSs}KdPub;
G18: DS possess at[4] H(Ksd1,TSs);
G19: DS know at[4] S send at[4] {idS, {Ksd1,TSs}KdPub,
     H(Ksd1,TSs)}KsPriv;
G20: DS know at[4] not(Zero possess at[0] {idS, {Ksd1,TSs}KdPub,
     H(Ksd1,TSs)}KsPriv);
G21: D possess at[5] idS;
G22: D possess at[5] Ksd1;
G23: D possess at[5] TSs;
G24: D know at[5] S possess at[5] Ksd1;
G25: D know at[5] S send at[5] {idS, {Ksd1,TSs}KdPub}KsPriv;
G26: D know at[5] not(Zero possess at[0] {idS, {Ksd1,TSs} KdPub}
     KsPriv);
G27: S possess at[6] idD;
G28: S possess at[6] H(Ksd1,TSs);
G29: S know at[6] D send at[6] {idD, H(Ksd1,TSs)}KdPriv;
G30: S know at[6] not(Zero possess at[0] {idD,H(Ksd1,TSs)}KdPriv);
```

```
G31: S possess at[7] instruc;
G32: S possess at[7] H(Ksd1, instruc, idS);
G33: S know at[7] D send at[7] {idD, idS, {instruc}Ksd1, H(Ksd1,
     instruc, idS)}KdPriv;
G34: S know at[7] not(Zero possess at[0] {idD, idS, {instruc}
     Ksd1, H(Ksd1, instruc, idS)}KdPriv);
G35: D possess at[8] message;
G36: D know at[8] S send at[8] {idS, idD, {message}Ksd1}KsPriv;
G37: D know at[8] not(Zero possess at[0] {idS, idD, {message}
     Ksd1} KsPriv);
```

Goals G1, G2, G3 states that DS possesses idS, {Ksd,TSs}KdPub and H(Ksd,TSs) after step 1 is completed. Goal G4 states DS knows S is the source of the message idS, {Ksd,TSs}KdPub, H(Ksd,TSs). G5 states DS also knows that this message is fresh, i.e. it has been created by S for the current protocol run.

G6, G7 and G8 states D possesses idS, Ksd and TSs after step 2 is completed. Goal G9 states D knows S possess at time t2 Ksd. G10 states that D knows DS is the source of the message idS, {Ksd,TSs}KdPub. G11 states that D also knows that this message is fresh, i.e. it has been created by DS for the current protocol run.

Goal G12 and G13 states that S possesses idD and H(Ksd,TSs) after step 3 is completed. Goal G14 states that S knows D is the source of the message idD, H(Ksd,TSs). G15 states that S also knows that this message is fresh, i.e. it has been created by D for the current protocol run.

G16, G17, G18 states DS possesses idS, {Ksd1,TSs}KdPub, H(Ksd1,TSs) after step 4 is completed. Goal G19 states that DS knows S is the source of the message idS, {Ksd1,TSs}KdPub, H(Ksd1,TSs). G20 states DS also knows that this message is fresh, i.e. it has been created by S for the current protocol run.

Goals G21, G22 and G23 states that D possesses idS and {Ksd1,TSs}KdPub respectively after step 5 is completed. G24 states D knows S possess at time t5 Ksd1. G25 states that D knows that the message idS, {Ksd1,TSs}KdPub was sent by S, i.e, D knows that S is certainly the source of this message. G26 states that D also knows that this message is fresh, i.e. it has been created by S for the current protocol run.

Goals G27, G28 states that S possesses idD, H(Ksd1,TSs) after step 6 is completed. G29 states that S knows that the message idD, H(Ksd1,TSs) was sent by D, i.e, S knows that D is certainly the source of this message. G30 states that S also knows that this message is fresh, i.e. it has been created by D for the current protocol run.

G31, G32 states that S possess instruct and H(Ksd1, instruc, idS) after step 7 is completed. G33 states that S knows that the message idD, idS, {instruc}Ksd1, H(Ksd1, instruc, idS) was sent by D, i.e, S knows that D is certainly the source of this message. G34 states that S also knows that this message is fresh, i.e. it has been created by D for the current protocol run.

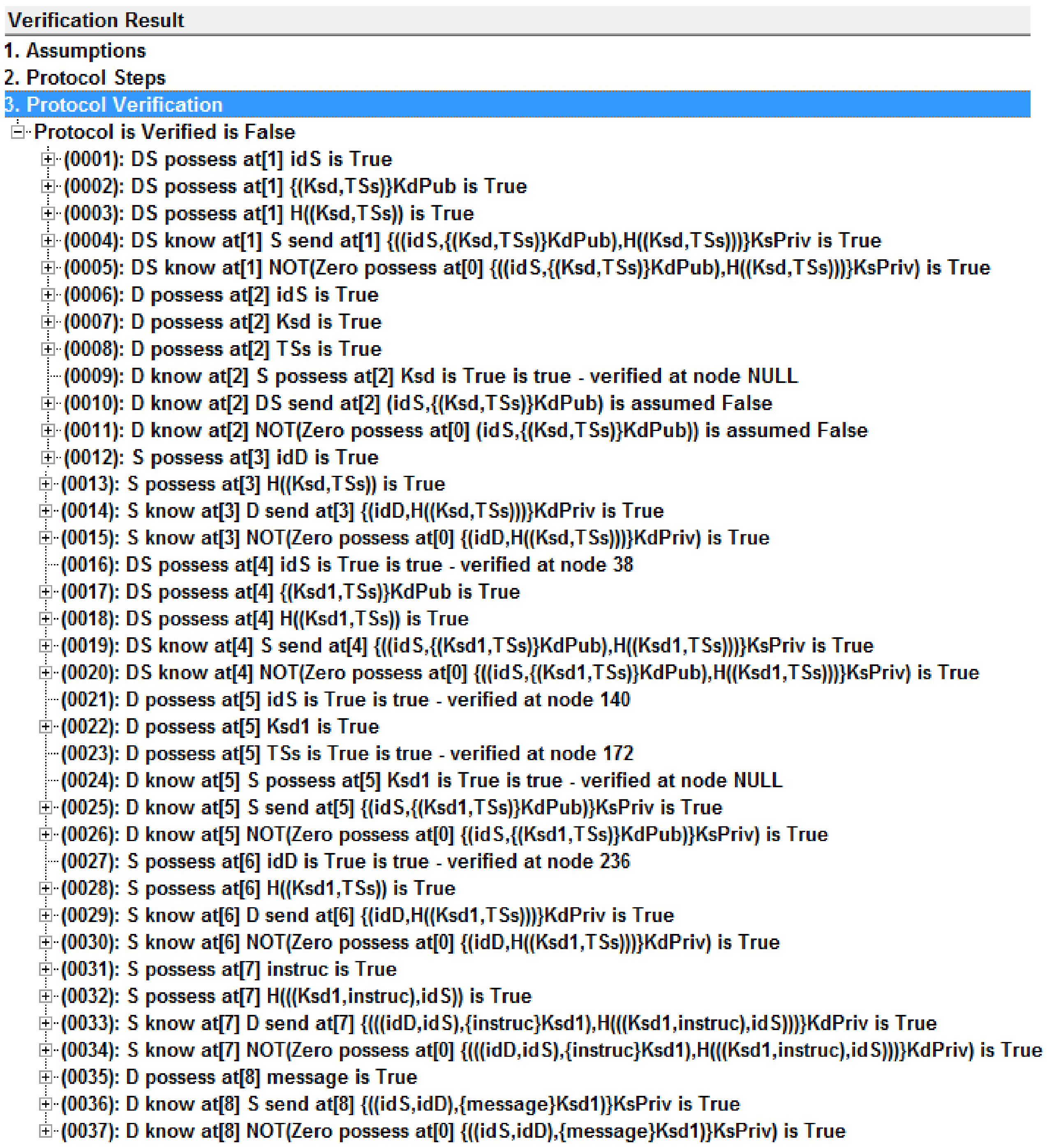
Goal G35 states that D possess message after step 8 is completed. G36 states that D knows that the message idS, idD, {message}Ksd1 was sent by S, i.e, D knows that S is certainly the source of this message. G37 states that D also knows that this message is fresh, i.e. it has been created by S for the current protocol run.

8.5.2 Verification results of patient-health monitoring WBAN in CDVT

The secure patient-health monitoring WBAN has been verified to be False in the CDVT tool. While all other goals have verified to be true, goals G10 and G11 have proven to be False.

Even though the protocol succeeds in authentication and verification of doctor and sensor, and successful transfer of session key, it falls short in completely securing the data sink. The failure of goal G10 points to doctor D not being able to prove that the message {Ksd,TSs}KdPub was forwarded by the data sink DS.

Similarly, failure of goal 11 refers to D’s inability to prove that no other entity possesses the message {Ksd,TSs}KdPub at that time. Therefore, the data sink must be secured in a more efficient way for the patient-health monitoring protocol to be securely carried out.



```
Verification Result
1. Assumptions
2. Protocol Steps
3. Protocol Verification
⊟ Protocol is Verified is False
   ⊞ (0001): DS possess at[1] idS is True
   ⊞ (0002): DS possess at[1] {(Ksd,TSs)}KdPub is True
   ⊞ (0003): DS possess at[1] H((Ksd,TSs)) is True
   ⊞ (0004): DS know at[1] S send at[1] {((idS,{(Ksd,TSs)}KdPub),H((Ksd,TSs)))}KsPriv is True
   ⊞ (0005): DS know at[1] NOT(Zero possess at[0] {((idS,{(Ksd,TSs)}KdPub),H((Ksd,TSs)))}KsPriv) is True
   ⊞ (0006): D possess at[2] idS is True
   ⊞ (0007): D possess at[2] Ksd is True
   ⊞ (0008): D possess at[2] TSs is True
     (0009): D know at[2] S possess at[2] Ksd is True is true - verified at node NULL
   ⊞ (0010): D know at[2] DS send at[2] (idS,{(Ksd,TSs)}KdPub) is assumed False
   ⊞ (0011): D know at[2] NOT(Zero possess at[0] (idS,{(Ksd,TSs)}KdPub)) is assumed False
   ⊞ (0012): S possess at[3] idD is True
   ⊞ (0013): S possess at[3] H((Ksd,TSs)) is True
   ⊞ (0014): S know at[3] D send at[3] {(idD,H((Ksd,TSs)))}KdPriv is True
   ⊞ (0015): S know at[3] NOT(Zero possess at[0] {(idD,H((Ksd,TSs)))}KdPriv) is True
     (0016): DS possess at[4] idS is True is true - verified at node 38
   ⊞ (0017): DS possess at[4] {(Ksd1,TSs)}KdPub is True
   ⊞ (0018): DS possess at[4] H((Ksd1,TSs)) is True
   ⊞ (0019): DS know at[4] S send at[4] {((idS,{(Ksd1,TSs)}KdPub),H((Ksd1,TSs)))}KsPriv is True
   ⊞ (0020): DS know at[4] NOT(Zero possess at[0] {((idS,{(Ksd1,TSs)}KdPub),H((Ksd1,TSs)))}KsPriv) is True
     (0021): D possess at[5] idS is True is true - verified at node 140
   ⊞ (0022): D possess at[5] Ksd1 is True
     (0023): D possess at[5] TSs is True is true - verified at node 172
     (0024): D know at[5] S possess at[5] Ksd1 is True is true - verified at node NULL
   ⊞ (0025): D know at[5] S send at[5] {(idS,{(Ksd1,TSs)}KdPub)}KsPriv is True
   ⊞ (0026): D know at[5] NOT(Zero possess at[0] {(idS,{(Ksd1,TSs)}KdPub)}KsPriv) is True
     (0027): S possess at[6] idD is True is true - verified at node 236
   ⊞ (0028): S possess at[6] H((Ksd1,TSs)) is True
   ⊞ (0029): S know at[6] D send at[6] {(idD,H((Ksd1,TSs)))}KdPriv is True
   ⊞ (0030): S know at[6] NOT(Zero possess at[0] {(idD,H((Ksd1,TSs)))}KdPriv) is True
   ⊞ (0031): S possess at[7] instruc is True
   ⊞ (0032): S possess at[7] H(((Ksd1,instruc),idS)) is True
   ⊞ (0033): S know at[7] D send at[7] {(((idD,idS),{instruc}Ksd1),H(((Ksd1,instruc),idS)))}KdPriv is True
   ⊞ (0034): S know at[7] NOT(Zero possess at[0] {(((idD,idS),{instruc}Ksd1),H(((Ksd1,instruc),idS)))}KdPriv) is True
   ⊞ (0035): D possess at[8] message is True
   ⊞ (0036): D know at[8] S send at[8] {((idS,idD),{message}Ksd1)}KsPriv is True
   ⊞ (0037): D know at[8] NOT(Zero possess at[0] {((idS,idD),{message}Ksd1)}KsPriv) is True
```

Figure 17: Verification result of patient-health monitoring WBAN in CDVT

8.6 P2P service security protocol in M2M environment

Peer-to-peer service security protocol [16] is implemented in M2M environment to provide security in wireless communication and perform mutual authentication of the participating entities. It uses hash functions, real-time secret agents and encryption key values to secure M2M communication. P2P security protocol also aims at protecting M2M communication against intruder attacks.

The notations used in the formal specification of P2P service protocol [16] are:

- Alice(A) : Communicating entity
- Bob(B) : Communicating entity
- DBServer(DB) : Database Server
- KdbPub : Public Key
- KdbPriv : Secret Key
- idA : Identity of Alive
- idB : Identity of Bob
- idDB : Identity of DBServer
- secretA : Secret of Alice
- secretB : Secret of Bob
- Ka, Kb : Session Keys
- TSa, TSb : Timestamps

The steps involved in this protocol are:

1. Alice -> Bob : H(BobID), TSa, {BobID, Secret(Alice), Ka}KdbPub
2. Bob -> DBServer : H(BobID), TSb, {AliceID, Secret(Bob), Kb}KdbPub, {BobID, Secret(Alice), Ka}KdbPub
3. DBServer -> Bob : H(idDB,BobID), Ka(+)Kb
4. Bob -> Alice : {BobID}{Ka}(+)BobID, H(idDB)
5. Alice -> Bob : H(Ka, AliceID), TSa

8.6.1 Formal Verification of P2P service security protocol

- **Initial Assumptions**

Initial assumptions are statements defining what each principal possesses and knows at the beginning of a protocol run.

The initial assumptions for P2P service security protocol is as follows:

```
A1: A possess at[0] Ka;
A2: A possess at[0] KdbPub;
A3: A possess at[0] secretA;
A4: A possess at[0] TSa;
A5: A possess at[0] idA;
A6: A know at[0] NOT(Zero possess at[0] secretA);
A7: A know at[0] NOT(Zero possess at[0] Ka);
A8: A know at[0] NOT(Zero possess at[0] TSa);
A9: A know at[0] NOT(Zero possess at[0] idA);
A10: B possess at[0] Kb;
A11: B possess at[0] KdbPub;
A12: B possess at[0] secretB;
A13: B possess at[0] TSb;
A14: B possess at[0] idB;
A15: B know at[0] DB possess at[0] KdbPriv;
A16: B know at[0] NOT(Zero possess at[0] secretB);
A17: B know at[0] NOT(Zero possess at[0] Kb);
A18: B know at[0] NOT(Zero possess at[0] TSb);
A19: B know at[0] NOT(Zero possess at[0] idA);
A20: B know at[0] NOT(Zero possess at[0] idB);
A21: DB possess at[0] KdbPub;
A22: DB possess at[0] KdbPriv;
A23: DB possess at[0] idDB;
A24: DB know at[0] A possess at[0] secretA;
```

```
A25: DB know at[0] B possess at[0] secretB;
A26: DB know at[0] NOT(Zero possess at[0] KdbPriv);
A27: DB know at[0] NOT(Zero possess at[0] idDB);
A28: DB know at[0] NOT(Zero possess at[0] secretA);
A29: DB know at[0] NOT(Zero possess at[0] secretB);
A30: (((DB know at[0] A possess at[0] secretA) AND (DB know at[0]
     NOT(Zero possess at[0] secretA))) IMPLY (DB know at[2] A
     send at[1] {idB, secretA, Ka}KdbPub));
A31: (((DB know at[0] B possess at[0] secretB) AND (DB know at[0]
     NOT(Zero possess at[0] secretB))) IMPLY (DB know at[2] B
     send at[1] {idA, secretB, Kb}KdbPub));
```

Assumptions A1, A2, A3, A4 and A5 states principal A possesses Ka, KdbPub, secretA, TSa and idA respectively. Assumptions A6, A7, A8, A9 represents the fact that before start of the protocol run, principal A is aware that no other principal possesses secretA, Ka, TSa, idA at that time.

Similarly, assumptions A10, A11, A12, A13, A14 represents the fact that principal B possesses Kb, KdbPub, secretB, TSb, idB respectively, at the beginning of the protocol run. A15 states B knows before the beginning of the protocol run, that DB possesses key KdbPriv.

A16, A17, A18, A19, A20 states before start of the protocol run, B is aware that no other principal possesses secretB, Kb, TSb, idA, idB at that time.

A21, A22, A23 states DB possesses KdbPub, KdbPriv, idDB. A24, A25 states that DB knows A possess secretA and B possess secret at the beginning of protocol run. A26 to A29 states before start of the protocol run, DB is aware that no other principal possesses KdbPriv, idDB, secretA, secretB at that time.

A30 implies that if DB know A possesses secretA and no other principal possesses secretA at that time, then DB knows it is A that has sent the data containing secretA. A31 implies that if B possesses secretB and no other principal possesses secretB at that time, then DB knows it is B that has sent the data containing secretB.

- **Protocol Steps**

The P2P service security protocol steps are formalized as follows:

```
S1: B receivefrom A at[1] H(idB), TSa, {idB, secretA, Ka}KdbPub;
S2: DB receivefrom B at[2] H(idB), TSb, {idA, secretB, Kb}KdbPub,
    {idB, secretA, Ka}KdbPub;
S3: B receivefrom DB at[3] H(idDB,idB), {Ka, Kb}KdbPriv;
S4: A receivefrom B at[4] {idB}Ka, idB, H(idDB);
S5: B receivefrom A at[5] H(Ka, idA), TSa;
```

Statement S1 states that B receives from principal A in step one (at time t1) a message, which contains hash of B's identity idB, timestamp TSa and - B's identity idB, secret of A secretA, A's session key Ka – encrypted using DB Server's public key.

S2 states that DB receives from B in step two (at time t2) a message, which contains hash of B's identity idB, timestamp TSb, A's identity idA, secret of B secretB, B's session key Kb – encrypted using DB Server's public key, and the message B received from A in the previous step.

S3 states B receives from DB at time t3 hash of DB's identity idDB and B's identity idB, session keys Ka and Kb. In S4, A receives from B at time t4 idB encrypted using session key Ka, B's identity idB and hash of idDB. S5 states B receives from A hash of session key Ka and idA, and timestamp TSa.

- **Protocol Goals**

The major objective of P2P service security protocol is to establish mutual authentication through Database Server, obtain integrity of messages through hash functions and anonymity of identity of users.

These objectives are specified in the goals as follows:

```
G1: B possess at[1] H(idB);
G2: B possess at[1] TSa;
```

```
G3: B possess at[1] {idB, secretA, Ka}KdbPub;
G4: B know at[1] not(Zero possess at[0] H(idB), TSa, {idB, secretA,
    Ka}KdbPub);
G5: DB possess at[2] H(idB);
G6: DB possess at[2] TSb;
G7: DB possess at[2] Kb;
G8: DB possess at[2] Ka;
G9: DB possess at[2] secretA;
G10: DB possess at[2] secretB;
G11: DB know at[2] A send at[1] {idB, secretA, Ka}KdbPub;
G12: DB know at[2] B send at[1] {idA, secretB, Kb}KdbPub;
G13: B possess at[3] H(idDB, idB);
G14: B possess at[3] {Ka, Kb}KdbPriv;
G15: B possess at[3] Ka;
G16: B know at[3] DB send at[3] {Ka, Kb}KdbPriv;
G17: B know at[3] NOT(Zero possess at[0] {Ka, Kb}KdbPriv);
G18: A possess at[4] H(idDB);
G19: A possess at[4] {idB}Ka;
G20: A possess at[4] idB;
G21: A know at[4] NOT(Zero possess at[0] {idB}Ka, idB, H(idDB));
G22: B possess at[5] H(Ka, idA);
G23: B possess at[5] TSa;
G24: B know at[5] NOT(Zero possess at[0] H(Ka, idA), TSa);
```

Goals G1, G2 and G3 states that B possesses H(idB), TSa and {idB, secretA, Ka}KdbPub at step 1. G4 states that B knows that the message H(idB), TSa, {idB, secretA, Ka}KdbPub is fresh, i.e. it has been created by A for the current protocol run.

Goals G5, G6, G7, G8, G9, G10 states that DB possesses H(idB), TSb, Kb, Ka, secretA and secretB at step 2. G11 states that DB knows A send {idB, secretA, Ka}KdbPub after step 2 is completed. G12 states that DB knows B send at step 2 {idA, secretB, Kb}KdbPub.

G13, G14, G15 states that B possess after step 3 H(idDB,idB), {Ka, Kb}KdbPriv, and Ka. G16 states that B knows DB is certainly the source of the message {Ka, Kb}KdbPriv. G17 states B knows that this message is fresh and has been created during the current protocol run. The above steps represent mutual authentication of A and B at DB.

Goal G18, G19, G20 states A possess after step 4 H(idDB), {idB}Ka, idB. G21 states A knows that this message is fresh and has been created during the current protocol run. G22, G23 states that B possess H(Ka, idA) and TSa. G24 states B knows that this message is fresh and has been created during the current protocol run.

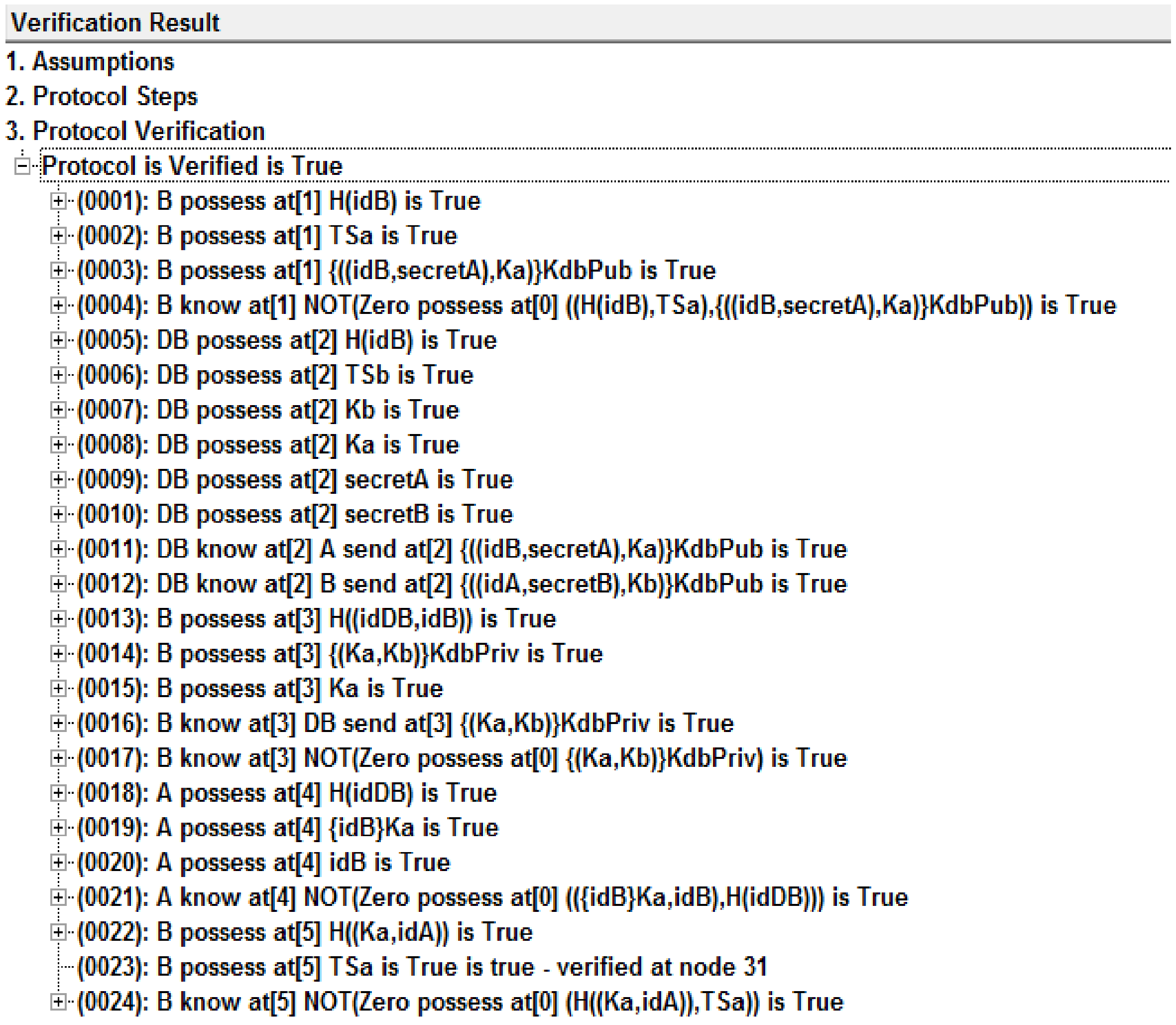
8.6.2 Verification results of P2P service security protocol in CDVT

The verification results of P2P service security protocol in M2M communication represents that the protocol is true, i.e, it satisfies all the goals and the tool found no weaknesses in the protocol.

Goals G11 and G12 are verified as true, leading to the fact that the authentication of principals A and B at DB is successful. At G11, DB knows that A certainly is the source of the message containing Ka and secretA and at G12, DB knows that B certainly is the source of the message containing Kb and secret. These represent mutual authentication were successful.

Goals G4, G17, G21, and G24 shows that no other principal possesses the messages mentioned in these goals, leading to the conclusion that the freshness of messages has been maintained throughout the course of the M2M communication.

Also, the use hash of identities instead of communicating identities directly, provides the feature of anonymity and privacy in M2M communication. Since all the messages are hash encrypted with values varying per session, M2M communication is safe against all intruder attacks as encrypted message cannot be decrypted.



```
Verification Result
1. Assumptions
2. Protocol Steps
3. Protocol Verification
⊟ Protocol is Verified is True
    ⊞ (0001): B possess at[1] H(idB) is True
    ⊞ (0002): B possess at[1] TSa is True
    ⊞ (0003): B possess at[1] {((idB,secretA),Ka)}KdbPub is True
    ⊞ (0004): B know at[1] NOT(Zero possess at[0] ((H(idB),TSa),{((idB,secretA),Ka)}KdbPub)) is True
    ⊞ (0005): DB possess at[2] H(idB) is True
    ⊞ (0006): DB possess at[2] TSb is True
    ⊞ (0007): DB possess at[2] Kb is True
    ⊞ (0008): DB possess at[2] Ka is True
    ⊞ (0009): DB possess at[2] secretA is True
    ⊞ (0010): DB possess at[2] secretB is True
    ⊞ (0011): DB know at[2] A send at[2] {((idB,secretA),Ka)}KdbPub is True
    ⊞ (0012): DB know at[2] B send at[2] {((idA,secretB),Kb)}KdbPub is True
    ⊞ (0013): B possess at[3] H((idDB,idB)) is True
    ⊞ (0014): B possess at[3] {(Ka,Kb)}KdbPriv is True
    ⊞ (0015): B possess at[3] Ka is True
    ⊞ (0016): B know at[3] DB send at[3] {(Ka,Kb)}KdbPriv is True
    ⊞ (0017): B know at[3] NOT(Zero possess at[0] {(Ka,Kb)}KdbPriv) is True
    ⊞ (0018): A possess at[4] H(idDB) is True
    ⊞ (0019): A possess at[4] {idB}Ka is True
    ⊞ (0020): A possess at[4] idB is True
    ⊞ (0021): A know at[4] NOT(Zero possess at[0] (({idB}Ka,idB),H(idDB))) is True
    ⊞ (0022): B possess at[5] H((Ka,idA)) is True
      (0023): B possess at[5] TSa is True is true - verified at node 31
    ⊞ (0024): B know at[5] NOT(Zero possess at[0] (H((Ka,idA)),TSa)) is True
```

Figure 18: Verification result of P2P service security protocol in CDVT

8.7 Secure communication in vehicular networks

A secure and lightweight communication protocol for vehicular networks [17] is an advancement over the Dynamic Key Distribution protocol for PKI-based VANETs, and is used to decrease the overhead related to secure transfer of information during vehicular communication. It describes a secure method for key pre-distribution and key establishment in vehicular networks. Every vehicle has a unique ID called electronic license plate (ELP) which is used in this protocol in the form of a key. Each vehicle has an on-board unit (OBU) and there is an RSU (Road-side unit) present at different places.

After the pre-distribution of the ELP key, both OBU and RSU calculate a shared key at its own end. This key is a function of coefficients of a polynomial and an ID from a pool id IDs. The communications specified in the protocol is between OBUs and RSUs.

The notations used in the formal specification of lightweight communication protocol in vehicular network are:

- OBU : On Board Units/Vehicle
- RSU : Road-side Units
- Kelp : Electronic license plate/Unique id of Vehicle
- KrsuPub : Public key of RSU
- KrsuPriv : Private key of RSU
- KobuPub : Public key of OBU
- KobuPriv : Private key of OBU
- polynomial : Coeffs of polynomial
- id : ID selected by RSU from pool of IDs for a particular OBU

The following are the steps involved in this protocol:

1. OBU -> RSU : {{Kelp}KrsuPub}KobuPriv
   RSU : {{{Kelp}KrsuPub}KobuPriv}KobuPub
   = {{Kelp}KrsuPub}KrsuPriv = Kelp
2. RSU -> OBU : {Coeffs of polynomial, id}Kelp
   OBU : {{Coeffs of polynomial, id}Kelp}Kelp
   = (Coeffs of polynomial, id)
   OBU : Shared key = F(Coeffs of polynomial, id)

8.7.1 Formal Verification of secure communication protocol in vehicular network

- **Initial Assumptions**

Initial assumptions are statements defining what each principal possesses and knows at the beginning of a protocol run.

The initial assumptions for vehicular network communication protocol is as follows:

```
A1: OBU possess at[0] Kelp;
A2: OBU possess at[0] KobuPub;
A3: OBU possess at[0] KobuPriv;
A4: OBU possess at[0] KrsuPub;
A5: OBU know at[0] NOT(Zero possess at[0] Kelp);
A6: OBU know at[0] NOT(Zero possess at[0] KobuPriv);
A7: OBU know at[0] RSU possess at[0] KobuPub;
A8: OBU know at[0] RSU possess at[0] polynomial;
A9: OBU know at[0] RSU possess at[0] id;
A10: RSU possess at[0] KrsuPub;
A11: RSU possess at[0] KrsuPriv;
A12: RSU possess at[0] polynomial;
A13: RSU possess at[0] id;
A14: RSU possess at[0] KobuPub;
A15: RSU know at[0] NOT(Zero possess at[0] KrsuPriv);
A16: RSU know at[0] OBU possess at[0] KrsuPub;
A17: RSU know at[0] OBU possess at[0] Kelp;
A18: RSU know at[0] OBU possess at[0] KobuPriv;
```

Assumptions A1, A2, A3 and A4 states that principal OBU possesses Kelp, KobuPub, KobuPriv and KrsuPub respectively. Assumptions A5 and A6 represents the fact that before start of the protocol run, OBU is aware that no other principal possesses Kelp and KobuPriv at that time. A7, A8, A9 states that OBU knows RSU possesses KobuPub, polynomial and id.

Similarly, assumptions A10, A11, A12, A13, A14 represents the fact that principal RSU possesses KrsuPub, KrsuPriv, polynomial, id, KobuPub respectively, at the beginning of the protocol run. A15 states before start of the protocol run, RSU is aware that no other principal possesses KrsuPriv at that time. A16, A17, A18 states RSU knows before the beginning of the protocol run, that OBU possesses KrsuPub, Kelp, KobuPriv.

- **Protocol Steps**

The steps of secure communication protocol for vehicular networks are formalized as follows:

```
S1: RSU receivefrom OBU at[1] {{Kelp}KrsuPub}KobuPriv;
S2: OBU receivefrom RSU at[2] {polynomial, id}Kelp;
```

Statement S1 states that RSU receives from principal OBU in step one (at time t1) a message, containing Kelp encrypted using KrsuPub. S2 states that OBU receives from RSU in step two (at time t2) a message, which contains coefficients of polynomial and id, encrypted using key Kelp.

- **Protocol Goals**

The major objective of lightweight communication protocol for vehicular networks [17] is to perform pre-distribution and establishment of keys in a secure manner.

These objectives are specified in the goals as follows:

```
G1: RSU possess at[1] Kelp;
G2: RSU know at[1] OBU possess at[1] Kelp;
G3: RSU know at[1] OBU send at[1] {{Kelp}KrsuPub}KobuPriv;
G4: RSU know at[1] NOT(Zero possess at[0] {{Kelp}KrsuPub}KobuPriv);
G5: OBU possess at[2] polynomial;
G6: OBU possess at[2] id;
G7: OBU know at[2] RSU possess at[2] polynomial;
G8: OBU know at[2] RSU possess at[2] id;
```

```
G9: OBU know at[2] RSU send at[2] {polynomial, id}Kelp;
G10: OBU know at[2] NOT(Zero possess at[0] {polynomial, id}Kelp);
```

Goal G1 states that RSU possesses Kelp after step 1 is completed. G2 states that RSU knows OBU possesses Kelp. G3 states that RSU knows OBU is certainly the source of the message {{Kelp}KrsuPub}KobuPriv. G4 states that RSU knows that this message is fresh, i.e. it has been created by OBU for the current protocol run.

Goals G5 and G6 states that OBU possesses polynomial and id after step 2 is completed. G7 and G8 states that OBU knows RSU possesses polynomial and id. G9 states that OBU knows RSU is certainly the source of the message {polynomial, id}Kelp. G10 states that OBU knows that this message is fresh, i.e. it has been created by RSU for the current protocol run.

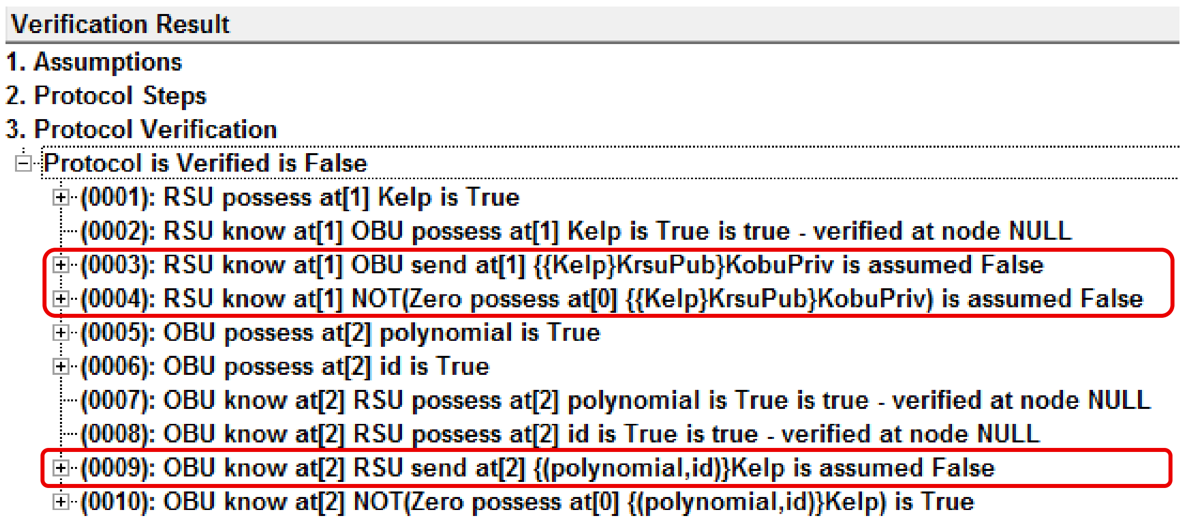
8.7.2 Verification results of secure vehicular communication protocol in CDVT

The lightweight communication protocol for vehicular networks has been verified to be False in the CDVT tool. While all other goals have verified to be true, goals G3, G4 and G9 have proven to be False.

The failure of these goals indicates that there is a possibility of a node being compromised, which in turn will lead to intruders obtaining access to key information. Goal G3 fails as RSU is not able to prove that the source of the message {Kelp}KrsuPub is OBU. Goal G9 fails as OBU is not able to prove that RSU sent the message {polynomial, id}Kelp. In these two cases, there is a possibility that it could be an intruder posing as OBU that sent the message.

Goal G4 fails as RSU is not able to prove that the message {Kelp}KrsuPub is fresh and that no other entity possesses the same message, i.e., there is no proof indicating the freshness of the message. Although, this protocol ensures confidentiality, these failed results show that there are possibilities for attackers to carry out replay attacks and man-in-the-middle (MITM) attack on this vehicular network.

These limitations can be overcome by using nonces during message transfer so as to provide a fresh element that will act as a proof that the message has not been replayed.



```
Verification Result
1. Assumptions
2. Protocol Steps
3. Protocol Verification
Protocol is Verified is False
    (0001): RSU possess at[1] Kelp is True
    (0002): RSU know at[1] OBU possess at[1] Kelp is True is true - verified at node NULL
    (0003): RSU know at[1] OBU send at[1] {{Kelp}KrsuPub}KobuPriv is assumed False
    (0004): RSU know at[1] NOT(Zero possess at[0] {{Kelp}KrsuPub}KobuPriv) is assumed False
    (0005): OBU possess at[2] polynomial is True
    (0006): OBU possess at[2] id is True
    (0007): OBU know at[2] RSU possess at[2] polynomial is True is true - verified at node NULL
    (0008): OBU know at[2] RSU possess at[2] id is True is true - verified at node NULL
    (0009): OBU know at[2] RSU send at[2] {(polynomial,id)}Kelp is assumed False
    (0010): OBU know at[2] NOT(Zero possess at[0] {(polynomial,id)}Kelp) is True
```

Figure 19: Verification result of secure vehicular network communication protocol in CDVT

8.8 TTP Based High-Efficient Multi-Key Exchange Protocol (THMEP)

THMEP [18] is a multiple key exchange protocol that has been implemented in various rounds of authentication and key exchange. The TTP acts as a trusted server and aids in key exchange between users A and B. THMEP claims to be safe against impersonation, eavesdropping, forgery and replay attacks as it uses hash functions and random parameters throughout the protocol.

The notations used in the formal specification of THMEP protocol are:

- A, B : Principals
- S : Trusted third party
- TSs : Timestamp of S
- Nsa, Nsb : Nonces of S to A and B
- idS, idA, idB : Identities of S, A and B
- Na, Nb : Nonces of A and B

The steps involved in the protocol are:

Round 1

1. S -> A : {IDs, TSs , s'A, SA}

   S : SA = {Nsa}KaPub

   s'A = F(H(Nsa),Kpwa)

2. S -> B : {IDs, TSs , s'B, SB}

   S : SB = {Nsb}KbPub

   s'B = F(H(Nsb),Kpwb)

Round 2

3. A -> S : {IDa; x'A; XA}

   A : XA = {Na}KsPub

   x'A = F(H(Na),Kpwa)

4. B -> S : {IDb; x'B; XB}

   B : XB = {Nb}KsPub

   x'B = F(H(Nb),Kpwb)

Round 3

5. S -> A : {IDs, IDb, XB, SB, nA}

   nA = H(IDa; IDs ; IDb; Na; Nsa; XB; SB; Kpwa)

6. S -> B : {IDs, IDa, XA, SA, nB}

   nB = H(IDb; IDs ; IDa; Nb; Nsb; XA; SA; Kpwb)

   A and B generate SK.

8.8.1 Formal Verification of THMEP protocol

- **Initial Assumptions**

Initial assumptions are statements defining what each principal possesses and knows at the beginning of a protocol run.

The initial assumptions for THMEP protocol is as follows:

```
A1: A possess at[0] KaPub;
A2: A possess at[0] KaPriv;
A3: A possess at[0] Na;
A4: A possess at[0] idA;
A5: A possess at[0] KttpPub;
A6: A possess at[0] KbPub;
A7: A know at[0] TTP possess at[0] KttpPriv;
A8: A know at[0] B possess at[0] KbPriv;
A9: A know at[0] NOT(Zero possess at[0] KaPriv);
A10: A know at[0] NOT(Zero possess at[0] Na);
A11: A know at[0] NOT(Zero possess at[0] idA);
A12: A know at[0] NOT(Zero possess at[0] idS);
A13: B possess at[0] KbPub;
A14: B possess at[0] KbPriv;
A15: B possess at[0] Nb;
A16: B possess at[0] idB;
A17: B possess at[0] KttpPub;
A18: B possess at[0] KaPub;
A19: B know at[0] TTP possess at[0] KttpPriv;
A20: B know at[0] A possess at[0] KaPriv;
A21: B know at[0] NOT(Zero possess at[0] KbPriv);
A22: B know at[0] NOT(Zero possess at[0] Nb);
A23: B know at[0] NOT(Zero possess at[0] idB);
A24: B know at[0] NOT(Zero possess at[0] idS);
A25: TTP possess at[0] KttpPub;
A26: TTP possess at[0] KttpPriv;
A27: TTP possess at[0] Nsa;
A28: TTP possess at[0] Nsb;
A29: TTP possess at[0] idS;
A30: TTP possess at[0] TSs;
```

```
A31: TTP possess at[0] Kpwa;
A32: TTP possess at[0] Kpwb;
A33: TTP possess at[0] KaPub;
A34: TTP possess at[0] KbPub;
A35: TTP know at[0] A possess at[0] KaPriv;
A36: TTP know at[0] B possess at[0] KbPriv;
A37: TTP know at[0] NOT(Zero possess at[0] KttpPriv);
A38: TTP know at[0] NOT(Zero possess at[0] idS);
```

Assumptions A1, A2, A3, A4, A5 and A6 states principal A possesses KaPub, KaPriv, Na, idA, KttpPub and KbPub respectively. A7 and A8 represents the fact that principal A knows TTP possesses KttpPriv and B possesses KbPriv, at the beginning of the protocol run. A9, A10, A11 and A12 represents the fact that principal A is aware that no other principal possesses KaPriv, Na, idA and idS at that time.

Similarly, A13 to A18 states B possesses KbPub, KbPriv, Nb, idB, KttpPub and KaPub respectively. A19 and A20 states B knows before the beginning of the protocol run, that TTP possesses KttpPriv and A possesses KaPriv. A21, A22, A23 and A24 states before start of the protocol run, B is aware that no other principal possesses KbPriv, Nb, idB and idS at that time.

A25 to A34 states TTP possesses KttpPub, KttpPriv, Nsa, Nsb, idS, TSs, Kpwa, Kpwb, KaPub and KbPub. A35 and A36 states TTP knows before the beginning of the protocol run, that A possesses KaPriv and B possesses KbPriv. A37 and A38 states before start of the protocol run, TTP is aware that no other principal possesses KttpPriv and idS at that time.

- **Protocol Steps**

The THMEP [18] protocol steps are formalized as follows:

```
S1: A receivefrom TTP at[1] {idS, TSs, F(H(Nsa), Kpwa), {Nsa}
    KaPub}KttpPriv;
```

```
S2: B receivefrom TTP at[2] {idS, TSs, F(H(Nsb), Kpwb), {Nsb}
    KbPub}KttpPriv;
S3: TTP receivefrom A at[3] {idA, F(H(Na), Kpwa), {Na}KttpPub}
    KaPriv;
S4: TTP receivefrom B at[4] {idB, F(H(Nb), Kpwb), {Nb}KttpPub}
    KbPriv;
S5: A receivefrom TTP at[5] {idS, idB, {Nb}KttpPub, {Nsb}KbPub,
    H(idA, idS, idB, Na, Nsa, {Nb}KttpPub, {Nsb}KbPub, Kpwa)}
    KttpPriv;
S6: B receivefrom TTP at[6] {idS, idA, {Na}KttpPub, {Nsa}KaPub,
    H(idB, idS, idA, Nb, Nsb, {Na}KttpPub, {Nsa}KaPub, Kpwb)}
    KttpPriv;
```

Statement S1 states that A receives from principal TTP in step one (at time t1) a message, which contains TTP's identity idS, timestamp TSs, a function of - hash of Nsa and Kpwa, and Nsa encrypted using A's public key.

S2 states that B receives from TTP in step two (at time t2) a message, which contains TTP's identity idS, timestamp TSs, a function of - hash of Nsb and Kpwb, and Nsb encrypted using B's public key.

S3 states TTP receives from A at time t3 A's identity idA, a function of - hash of Na and Kpwa, and Na encrypted using TTP's public key. In S4, TTP receives from B at time t4 B's identity idB, a function of - hash of Nb and Kpwb, and Nb encrypted using TTP's public key.

S5 states A receives from TTP at time t5 idS, idB, Nb encrypted using KttpPub, Nsb encrypted using KbPub, hash of - idA, idS, idB, Na, Nsa, Nb encrypted using KttpPub, Nsb encrypted using KbPub and Kpwa.

S6 states B receive from TTP at time t6 idS, idA, Na encrypted using KttpPub, Nsa encrypted using KaPub, hash of - idB, idS, idA, Nb, Nsb, Na encrypted using KttpPub, Nsa encrypted using KaPub and Kpwb.

- **Protocol Goals**

The major objective of THMEP protocol is to establish mutual authentication between users and TTP and to resist against replay, eavesdropping, known-key, impersonation, and forgery attacks.

These objectives are specified in the goals as follows:

```
G1: A possess at[1] F(H(Nsa), Kpwa);
G2: A possess at[1] Nsa;
G3: A possess at[1] idS;
G4: A know at[1] TTP send at[1] {idS, TSs, F(H(Nsa), Kpwa), {Nsa}
     KaPub}KttpPriv;
G5: B possess at[2] F(H(Nsb), Kpwb);
G6: B possess at[2] Nsb;
G7: B possess at[2] idS;
G8: B know at[2] TTP send at[2] {idS, TSs, F(H(Nsb), Kpwb), {Nsb}
     KbPub}KttpPriv;
G9: TTP possess at[3] F(H(Na), Kpwa);
G10: TTP possess at[3] Na;
G11: TTP possess at[3] idA;
G12: TTP know at[3] A send at[3] {idA, F(H(Na), Kpwa), {Na}
     KttpPub}KaPriv;
G13: TTP possess at[4] F(H(Nb), Kpwb);
G14: TTP possess at[4] Nb;
G15: TTP possess at[4] idB;
G16: TTP know at[4] B send at[4] {idB, F(H(Nb), Kpwb), {Nb}
     KttpPub}KbPriv;
G17: A possess at[5] idB;
G18: A possess at[5] H(idA, idS, idB, Na, Nsa, {Nb}KttpPub, {Nsb}
     KbPub, Kpwa);
G19: A know at[5] TTP send at[5] {idS, idB, {Nb}KttpPub, {Nsb}
     KbPub, H(idA, idS, idB, Na, Nsa, {Nb}KttpPub, {Nsb}KbPub,
     Kpwa)}KttpPriv;
```

```
G20: B possess at[6] idA;
G21: B possess at[6] H(idB, idS, idA, Nb, Nsb, {Na}KttpPub, {Nsa}
     KaPub, Kpwb);
G22: B know at[6] TTP send at[6] {idS, idA, {Na}KttpPub, {Nsa}
     KaPub, H(idB, idS, idA, Nb, Nsb, {Na}KttpPub, {Nsa}KaPub,
     Kpwb)}KttpPriv;
```

Goals G1, G2 and G3 states that A possesses F(H(Nsa), Kpwa), Nsa, and idS at step 1. G4 states that A knows that the message {idS, TSs, F(H(Nsa), Kpwa), {Nsa}KaPub}KttpPriv was sent by TTP in step 1. Similarly, Goals G5, G6 and G7 states that B possesses F(H(Nsb), Kpwb), Nsb, and idS at step 2. G8 states that B knows that the message {idS, TSs, F(H(Nsb), Kpwb), {Nsb}KbPub}KttpPriv was sent by TTP in step 2.

G9, G10 and G11 states TTP possesses F(H(Na), Kpwa), Na, and idA after step 3 is complete. G12 states that TTP knows that the message {idA, F(H(Na), Kpwa), {Na}KttpPub}KaPriv was sent by A in step 3. G13, G14 and G15 states TTP possesses F(H(Nb), Kpwb), Nb, and idB after step 4 is complete. G16 states that TTP knows that the message {idB, F(H(Nb), Kpwb), {Nb}KttpPub}KbPriv was sent by B in step 4.

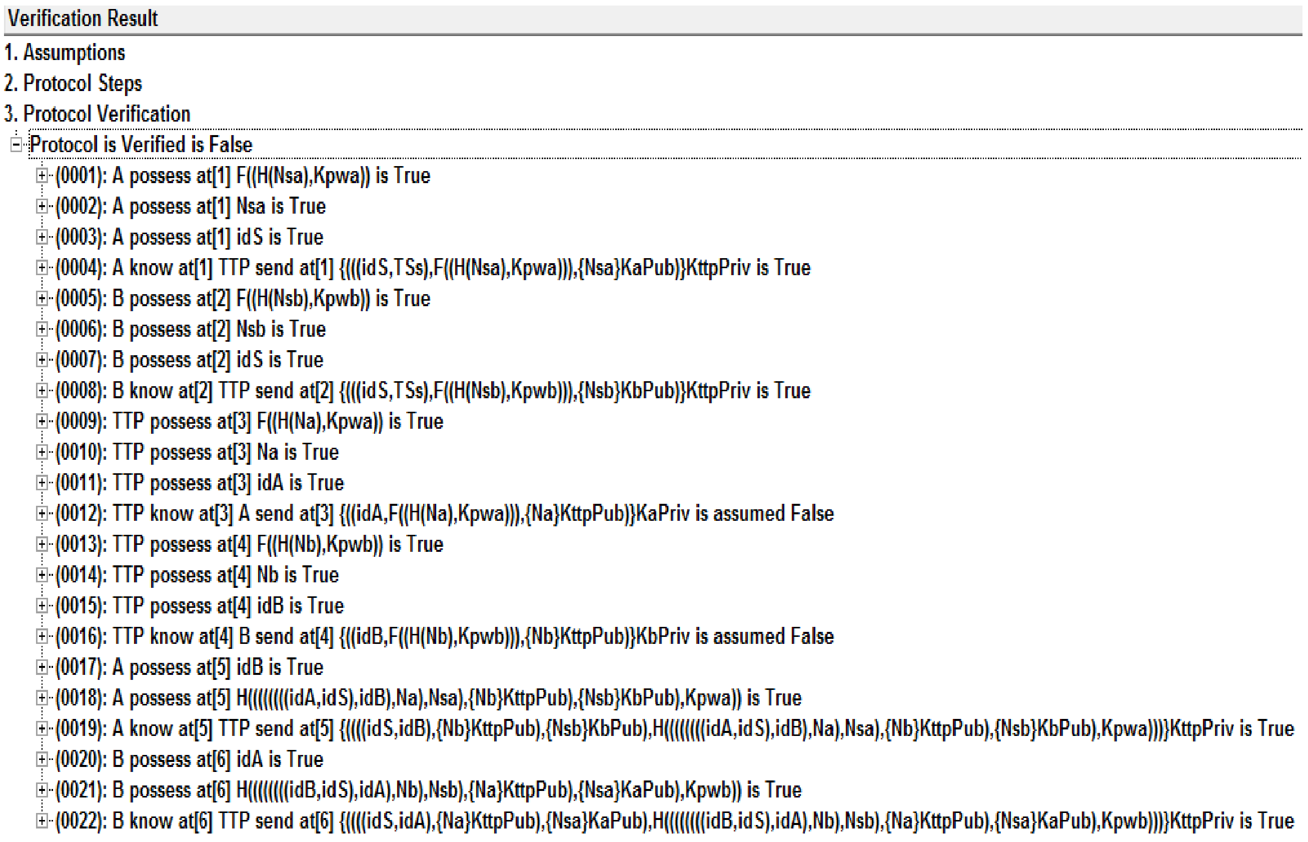
G17 and G18 states A possesses idB and H(idA, idS, idB, Na, Nsa, {Nb}KttpPub, {Nsb}KbPub, Kpwa) after step 5 is complete. G19 states A knows at step 5 TTP is the source of the message {idS, idB, {Nb}KttpPub, {Nsb}KbPub, H(idA, idS, idB, Na, Nsa, {Nb}KttpPub, {Nsb}KbPub, Kpwa)}KttpPriv. G20 and G21 states B possesses idA and H(idB, idS, idA, Nb, Nsb, {Na}KttpPub, {Nsa}KaPub, Kpwb) after step 6 is complete. G22 states B knows at step 6 TTP is the source of the message {idS, idA, {Na}KttpPub, {Nsa}KaPub, H(idB, idS, idA, Nb, Nsb, {Na}KttpPub, {Nsa}KaPub, Kpwb)}KttpPriv.

8.8.2 Verification results of THMEP protocol in CDVT

The verification results of THMEP [18] protocol has been verified to be False in the CDVT tool. While all other goals have verified to be true, goals G12 and G16 have proven to be False.

The failure of these goals indicates that there is a possibility for an adversary to act as a genuine user and obtain session keys by guessing passwords chosen by the users, i.e, THMEP is vulnerable to password-guessing attack. For instance, an intruder posing as user A, after receiving the initial message from TTP, can block the communication and can guess the password pwA from the password space D, and calculate password key Kpwa from pwA. It can then form SA and s'A and compare these values to the one it obtained from TTP. If both are same, then the password pwA the intruder has guessed is correct and the attack is successful.

Therefore, goal G12 fails as TTP is not able to prove that the message idA, F(H(Na), Kpwa), {Na}KttpPub was sent by A. G16 fails as TTP is not able to prove that the message idB, F(H(Nb), Kpwb), {Nb}KttpPub was sent by B. Therefore, contrary to its claims, this protocol is vulnerable to replay, man-in-the-middle and offline password-guessing attacks.



```
Verification Result
1. Assumptions
2. Protocol Steps
3. Protocol Verification
 ⊟ Protocol is Verified is False
    ⊞ (0001): A possess at[1] F((H(Nsa),Kpwa)) is True
    ⊞ (0002): A possess at[1] Nsa is True
    ⊞ (0003): A possess at[1] idS is True
    ⊞ (0004): A know at[1] TTP send at[1] {(((idS,TSs),F((H(Nsa),Kpwa))),{Nsa}KaPub)}KttpPriv is True
    ⊞ (0005): B possess at[2] F((H(Nsb),Kpwb)) is True
    ⊞ (0006): B possess at[2] Nsb is True
    ⊞ (0007): B possess at[2] idS is True
    ⊞ (0008): B know at[2] TTP send at[2] {(((idS,TSs),F((H(Nsb),Kpwb))),{Nsb}KbPub)}KttpPriv is True
    ⊞ (0009): TTP possess at[3] F((H(Na),Kpwa)) is True
    ⊞ (0010): TTP possess at[3] Na is True
    ⊞ (0011): TTP possess at[3] idA is True
    ⊞ (0012): TTP know at[3] A send at[3] {((idA,F((H(Na),Kpwa))),{Na}KttpPub)}KaPriv is assumed False
    ⊞ (0013): TTP possess at[4] F((H(Nb),Kpwb)) is True
    ⊞ (0014): TTP possess at[4] Nb is True
    ⊞ (0015): TTP possess at[4] idB is True
    ⊞ (0016): TTP know at[4] B send at[4] {((idB,F((H(Nb),Kpwb))),{Nb}KttpPub)}KbPriv is assumed False
    ⊞ (0017): A possess at[5] idB is True
    ⊞ (0018): A possess at[5] H(((((((idA,idS),idB),Na),Nsa),{Nb}KttpPub),{Nsb}KbPub),Kpwa)) is True
    ⊞ (0019): A know at[5] TTP send at[5] {((((idS,idB),{Nb}KttpPub),{Nsb}KbPub),H(((((((idA,idS),idB),Na),Nsa),{Nb}KttpPub),{Nsb}KbPub),Kpwa)))}KttpPriv is True
    ⊞ (0020): B possess at[6] idA is True
    ⊞ (0021): B possess at[6] H(((((((idB,idS),idA),Nb),Nsb),{Na}KttpPub),{Nsa}KaPub),Kpwb)) is True
    ⊞ (0022): B know at[6] TTP send at[6] {((((idS,idA),{Na}KttpPub),{Nsa}KaPub),H(((((((idB,idS),idA),Nb),Nsb),{Na}KttpPub),{Nsa}KaPub),Kpwb)))}KttpPriv is True
```

Figure 20: Verification result of THMEP protocol in CDVT

8.9 MAKA scheme with user anonymity in GLOMONET

Mutual Authentication and Key Agreement (MAKA) scheme [19] provides mutual authentication and user anonymity for roaming devices in GLOMONETs. This scheme is based on hash functions and XOR operations and establishes a session key. Mobile user MS and foreign agent FA authenticates each other through home agent, HA. The three phases of MAKA scheme are registration phase, mutual authentication and key agreement phase, and password renewal phase.

The notations used in the formal specification of MAKA scheme [19] are:

- Ms : Mobile Station
- B : Foreign Agent
- C : Home Agent
- idM : Identity of Mobile Station
- idF : Identity of Foreign Agent
- idH : Identity of Home Agent
- Nm : Nonce generated by Mobile user
- Nh : Nonce generated by Home Agent
- Kuh : Key generated by Home Agent
- pid : Psuedo-identity of M
- Kem : Emergency key
- seq : Transaction sequence (increments by 1)
- Nf : Nonce generated by Foreign Agent
- Kfh : Shared key between B and C

The steps involved in the protocol are:

Phase 1 - Registration

1. Ms -> C : idM

   C : Nh, Kuh = H(idM,Nh)
2. C -> Ms : Kuh, (pid, Kem), seq

Phase 2 - MAKA Phase

3. Ms -> B : AidM, Nx, seq, IDh
4. Ms: AidM = H(idM, Kuh, Nm, seq)

   Nx = F(H(idM, Kuh), Nm)
5. B -> C : AidM, Nx, seq, IDf, V1, Ny

   B : Ny = F(H(Kfh), Nf)

   V1 = H(AidM, Nx, seq, IDh, Ny, Kfh, Nf)
6. C -> B : N’x, N’y, V2, V3, Ts

   C : N'x = F(H(Kuh, idM, seq), Nf)

   N'y = F(H(Kf, Nf), Nm)

   V2 = F(H(N'y, Nf), Kfh)

   Ts = F(H(Kuh, idM, Nm), seq)

   V3 = F(H(N'x, Nm, Ts), Kuh)
7. B -> Ms : N’x, V3, Ts

8.9.1 Formal Verification of MAKA scheme

- **Initial Assumptions**

Initial assumptions are statements defining what each principal possesses and knows at the beginning of a protocol run.

The initial assumptions for MAKA scheme is as follows:

```
A1: Ms possess at[0] idM;
A2: Ms possess at[0] idH;
A3: Ms possess at[0] Nm;
A4: Ms possess at[0] PSW;
A5: Ms know at[0] NOT(Zero possess at[0] idM);
A6: Ms know at[0] NOT(Zero possess at[0] Nm);
A7: Ms know at[0] NOT(Zero possess at[0] PSW);
```

```
A8: Ms know at[0] B possess at[0] Nf;
A9: B possess at[0] idF;
A10: B possess at[0] Nf;
A11: B possess at[0] Kfh;
A12: B know at[0] Ms possess at[0] PSW;
A13: B know at[0] NOT(Zero possess at[0] idF);
A14: B know at[0] NOT(Zero possess at[0] Nf);
A15: B know at[0] NOT(Zero possess at[0] Kfh);
A16: B know at[0] C possess at[0] Nh;
A17: C possess at[0] idH;
A18: C possess at[0] pid;
A19: C possess at[0] Kem;
A20: C possess at[0] seq;
A21: C possess at[0] Nh;
A22: C know at[0] NOT(Zero possess at[0] Nh);
A23: C know at[0] NOT(Zero possess at[0] pid);
A24: C know at[0] NOT(Zero possess at[0] Kem);
A25: C know at[0] B possess at[0] Kfh;
A26: B know at[3] ((B receivefrom Ms at[3] H(idM, F(H(idM, Nh),
     H(idM, PSW)), Nm, seq), F(H(idM, H(idM, Nh)), Nm), seq, idH
     AND Ms possess at[0] PSW) IMPLY (Ms send at[3] H(idM,
     F(H(idM, Nh), H(idM, PSW)), Nm, seq), F(H(idM, H(idM, Nh)),
     Nm)));
A27: C know at[4] ((C receivefrom B at[4] H(idM, H(idM, Nh), Nm,
     seq), F(H(idM, H(idM, Nh)), Nm), seq, idF, H(H(idM, H(idM,
     Nh), Nm, seq), F(H(idM, H(idM, Nh)), Nm), seq, idH,
     F(H(Kfh), Nf), Kfh, Nf), F(H(Kfh), Nf) AND B possess at[0]
     Kfh) IMPLY (B send at[4] H(H(idM, H(idM, Nh), Nm, seq),
     F(H(idM, H(idM, Nh)), Nm), seq, idH, F(H(Kfh), Nf), Kfh,
     Nf)));
A28: B know at[5] ((B receivefrom C at[5] F(H(H(idM, Nh), idM,
     seq), Nf), F(H(Kfh, Nf), Nm), F(H(F(H(Kfh, Nf), Nm), Nf),
     Kfh), F(H(F(H(H(idM, Nh), idM, seq), Nf), Nm, F(H(H(idM,
     Nh), idM, Nm), seq)), H(idM, Nh)), F(H(H(idM, Nh), idM, Nm),
     seq) AND C possess at[0] Nh) IMPLY (C send at[5]
     F(H(F(H(H(idM, Nh), idM, seq), Nf), Nm, F(H(H(idM, Nh), idM,
     Nm), seq)), H(idM, Nh))));
A29: Ms know at[6] ((Ms receivefrom B at[6] F(H(H(idM, Nh), idM,
     seq), Nf), F(H(F(H(H(idM, Nh), idM, seq), Nf), Nm,
     F(H(H(idM, Nh), idM, Nm), seq)), H(idM, Nh)), F(H(H(idM,
     Nh), idM, Nm), seq) AND B possess at[0] Nf) IMPLY (B send
     at[6] F(H(H(idM, Nh), idM, seq), Nf), F(H(F(H(H(idM, Nh),
     idM, seq), Nf), Nm, F(H(H(idM, Nh), idM, Nm), seq)), H(idM,
     Nh)), F(H(H(idM, Nh), idM, Nm), seq)));
```

Assumptions A1, A2, A3 and A4 states principal Ms possesses idM, idH, Nm and PSW respectively. A5, A6, A7 represents the fact that before start of the protocol run, principal Ms is aware that no other principal possesses idM, Nm, PSW at that time. A8 states that Ms knows B possesses Nf. Similarly, assumptions A9, A10, A11 represents the fact that principal B possesses idF, Nf, Kfh respectively, at the beginning of the protocol run. A12 states B knows Ms possesses PSW. A13, A14, A15 states before start of the protocol run, B is aware that no other principal possesses idF, Nf, Kfh at that time. A16 states that B knows C possesses Nh. A17, A18, A19, A20, A21 states C possesses idH, pid, Kem, seq and Nh. A22, A23, A24 states that C knows no other principal possesses Nh, pid, Kem at the beginning of protocol run. A25 states C know B possesses Kfh.

A26 states B knows if B receives from Ms H(idM, F(H(idM, Nh), H(idM, PSW)), Nm, seq), F(H(idM, H(idM, Nh)), Nm), seq, idH and Ms possesses PSW implies that Ms sent the message. A27 states C knows if C receive from B H(idM, H(idM, Nh), Nm, seq), F(H(idM, H(idM, Nh)), Nm), seq, idF, H(H(idM, H(idM, Nh), Nm, seq), F(H(idM, H(idM, Nh)), Nm), seq, idH, F(H(Kfh), Nf), Kfh, Nf), F(H(Kfh), Nf) and B possess Kfh, then B is the source of the message. A28 states B knows if B receives from C F(H(H(idM, Nh), idM, seq), Nf), F(H(Kfh, Nf), Nm), F(H(F(H(Kfh, Nf), Nm), Nf), Kfh), F(H(F(H(H(idM, Nh), idM, seq), Nf), Nm, F(H(H(idM, Nh), idM, Nm), seq)), H(idM, Nh)), F(H(H(idM, Nh), idM, Nm), seq and C possess Nh, then C has sent the message. A29 states Ms knows if Ms receives from B F(H(H(idM, Nh), idM, seq), Nf), F(H(F(H(H(idM, Nh), idM, seq), Nf), Nm, F(H(H(idM, Nh), idM, Nm), seq)), H(idM, Nh)), F(H(H(idM, Nh), idM, Nm), seq and B possess Nf, then B has sent the message.

- **Protocol Steps**

The MAKA scheme steps are formalized as follows:

```
S1: C receivefrom Ms at[1] idM;
S2: Ms receivefrom C at[2] H(idM, Nh), pid, Kem, seq;
S3: B receivefrom Ms at[3] H(idM, F(H(idM, Nh), H(idM, PSW)), Nm,
    seq), F(H(idM, H(idM, Nh)), Nm), seq, idH;
S4: C receivefrom B at[4] H(idM, H(idM, Nh), Nm, seq), F(H(idM,
    H(idM, Nh)), Nm), seq, idF, H(H(idM, H(idM, Nh), Nm, seq),
    F(H(idM, H(idM, Nh)), Nm), seq, idH, F(H(Kfh), Nf), Kfh, Nf),
    F(H(Kfh), Nf);
S5: B receivefrom C at[5] F(H(H(idM, Nh), idM, seq), Nf), F(H(Kfh,
    Nf), Nm), F(H(F(H(Kfh, Nf), Nm), Nf), Kfh), F(H(F(H(H(idM, Nh),
    idM, seq), Nf), Nm, F(H(H(idM, Nh), idM, Nm), seq)), H(idM,
    Nh)), F(H(H(idM, Nh), idM, Nm), seq);
S6: Ms receivefrom B at[6] F(H(H(idM, Nh), idM, seq), Nf),
    F(H(F(H(H(idM, Nh), idM, seq), Nf), Nm, F(H(H(idM, Nh), idM,
    Nm), seq)), H(idM, Nh)), F(H(H(idM, Nh), idM, Nm), seq);
```

Statement S1 states that C receives from principal Ms in step one (at time t1) a message, which contains the identity of Ms, idM. S2 states that Ms receives from C in step two (at time t2) a message, which contains H(idM, Nh), pid, Kem and seq. S3 states B receives from Ms at time t3 hash of - idM, F(H(idM, Nh), H(idM, PSW)), Nm, seq - F(H(idM, H(idM, Nh)), Nm), seq and idH.

In S4, C receives from B at time t4 H(idM, H(idM, Nh), Nm, seq), F(H(idM, H(idM, Nh)), Nm), seq, idF, H(H(idM, H(idM, Nh), Nm, seq), F(H(idM, H(idM, Nh)), Nm), seq, idH, F(H(Kfh), Nf), Kfh, Nf), F(H(Kfh), Nf). S5 states B receives from C F(H(H(idM, Nh), idM, seq), Nf), F(H(Kfh, Nf), Nm), F(H(F(H(Kfh, Nf), Nm), Nf), Kfh), F(H(F(H(H(idM, Nh), idM, seq), Nf), Nm, F(H(H(idM, Nh), idM, Nm), seq)), H(idM, Nh)), F(H(H(idM, Nh), idM, Nm), seq) at t5. S6 states Ms receives from B at time t6 F(H(H(idM, Nh), idM, seq), Nf), F(H(F(H(H(idM, Nh), idM, seq), Nf), Nm, F(H(H(idM, Nh), idM, Nm), seq)), H(idM, Nh)), F(H(H(idM, Nh), idM, Nm), seq).

- **Protocol Goals**

The major objective of MAKA scheme in GLOMONET preserve the privacy of user or user anonymity and mutual authentication between mobile user and foreign agent.

These objectives are specified in the goals as follows:

```
G1: C possess at[1] idM;
G2: Ms possess at[2] pid;
G3: Ms possess at[2] Kem;
G4: Ms possess at[2] seq;
G5: Ms possess at[2] H(idM, Nh);
G6: B possess at[3] H(idM, F(H(idM, Nh), H(idM, PSW)), Nm, seq);
G7: B possess at[3] F(H(idM, H(idM, Nh)), Nm);
G8: B possess at[3] seq;
G9: B possess at[3] idH;
G10: B know at[3] Ms send at[3] H(idM, F(H(idM, Nh), H(idM, PSW)),
     Nm, seq), F(H(idM, H(idM, Nh)), Nm);
G11: C possess at[4] H(idM, H(idM, Nh), Nm, seq);
G12: C possess at[4] F(H(idM, H(idM, Nh)), Nm), seq, idF;
G13: C possess at[4] H(H(idM, H(idM, Nh), Nm, seq), F(H(idM, H(idM,
     Nh)), Nm), seq, idH, F(H(Kfh), Nf), Kfh, Nf);
G14: C possess at[4] F(H(Kfh), Nf);
G15: C know at[4] B send at[4] H(H(idM, H(idM, Nh), Nm, seq),
     F(H(idM, H(idM, Nh)), Nm), seq, idH, F(H(Kfh), Nf), Kfh, Nf);
G16: B possess at[5] F(H(H(idM, Nh), idM, seq), Nf);
G17: B possess at[5] F(H(Kfh, Nf), Nm);
G18: B possess at[5] F(H(F(H(Kfh, Nf), Nm), Nf), Kfh);
G19: B possess at[5] F(H(F(H(H(idM, Nh), idM, seq), Nf), Nm,
     F(H(H(idM, Nh), idM, Nm), seq)), H(idM, Nh));
G20: B possess at[5] F(H(H(idM, Nh), idM, Nm), seq);
G21: B know at[5] C send at[5] F(H(F(H(H(idM, Nh), idM, seq), Nf),
     Nm, F(H(H(idM, Nh), idM, Nm), seq)), H(idM, Nh));
```

```
G22: Ms possess at[6] F(H(H(idM, Nh), idM, seq), Nf);
G23: Ms possess at[6] F(H(F(H(H(idM, Nh), idM, seq), Nf), Nm,
     F(H(H(idM, Nh), idM, Nm), seq)), H(idM, Nh));
G24: Ms possess at[6] F(H(H(idM, Nh), idM, Nm), seq);
G25: Ms know at[6] B send at[6] F(H(H(idM, Nh), idM, seq), Nf),
     F(H(F(H(H(idM, Nh), idM, seq), Nf), Nm, F(H(H(idM, Nh), idM,
     Nm), seq)), H(idM, Nh)), F(H(H(idM, Nh), idM, Nm), seq);
```

Goals G1 states that C possesses idM at step 1. G2, G3, G5 states that Ms possesses pid, Kem, seq, H(idM, Nh). G6, G7, G8, G9 states that B possesses after step 2 H(idM, F(H(idM, Nh), H(idM, PSW)), Nm, seq), F(H(idM, H(idM, Nh)), Nm), seq and idH respectively. G10 states that B knows Ms sent the message H(idM, F(H(idM, Nh), H(idM, PSW)), Nm, seq), F(H(idM, H(idM, Nh)), Nm) after step 3 is complete. G11, G12, G13, G14 states that C possesses H(idM, H(idM, Nh), Nm, seq), F(H(idM, H(idM, Nh)), Nm), seq, idF, H(H(idM, H(idM, Nh), Nm, seq), F(H(idM, H(idM, Nh)), Nm), seq, idH, F(H(Kfh), Nf), Kfh, Nf) and F(H(Kfh), Nf). G15 states that C know B send at step 4 the message H(H(idM, H(idM, Nh), Nm, seq), F(H(idM, H(idM, Nh)), Nm), seq, idH, F(H(Kfh), Nf), Kfh, Nf).

Goals G16 to G20 states that B possesses F(H(H(idM, Nh), idM, seq), Nf), F(H(Kfh, Nf), Nm), F(H(F(H(Kfh, Nf), Nm), Nf), Kfh) and F(H(F(H(H(idM, Nh), idM, seq), Nf), Nm, F(H(H(idM, Nh), idM, Nm), seq)), H(idM, Nh)) and F(H(H(idM, Nh), idM, Nm), seq) at step 5. G21 states that B knows C has sent the above message. G22, G23, G24 states that Ms possesses F(H(H(idM, Nh), idM, seq), Nf) and F(H(F(H(H(idM, Nh), idM, seq), Nf), Nm, F(H(H(idM, Nh), idM, Nm), seq)), H(idM, Nh)) and F(H(H(idM, Nh), idM, Nm), seq) after step 6 is complete. G25 states that Ms knows B has sent this message.

8.9.2 Verification results of MAKA scheme in CDVT

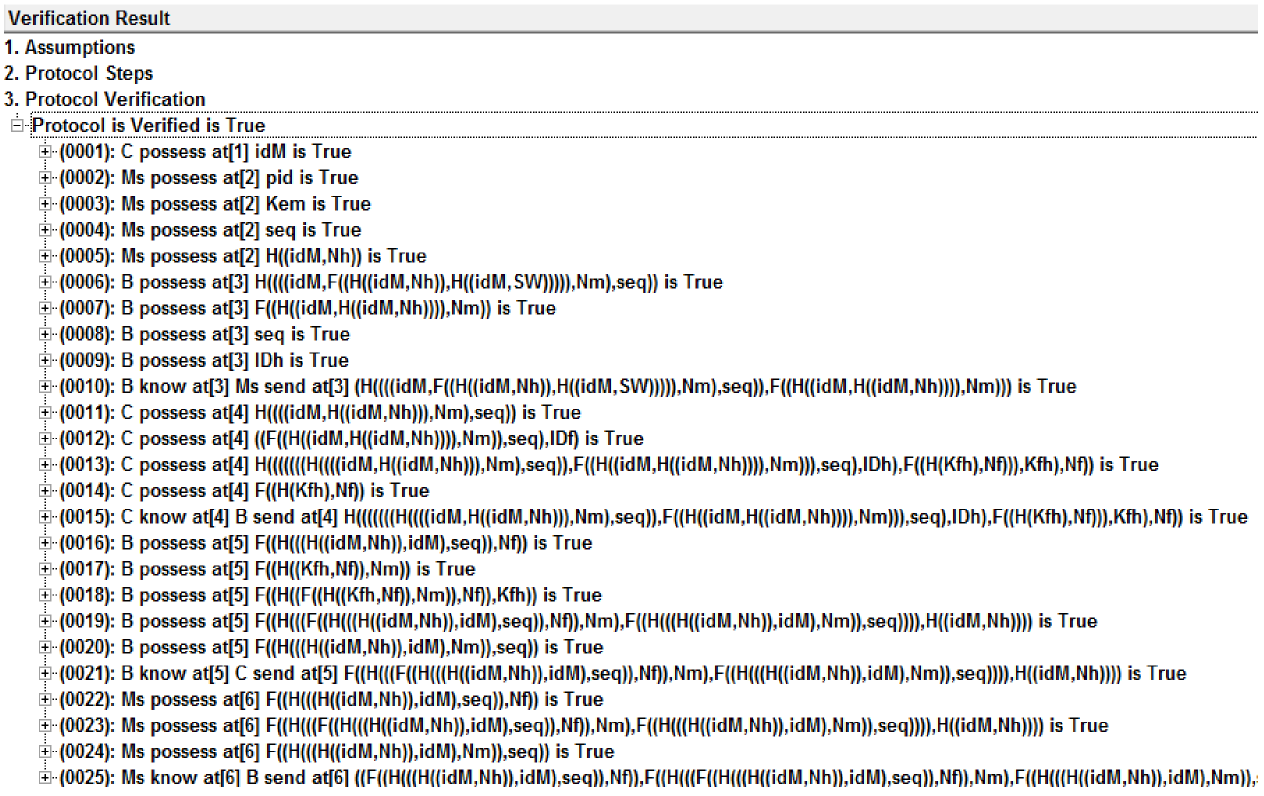
The verification results of MAKA scheme [19] in GLOMONETs represents that the protocol is true, i.e, it satisfies all the goals and the tool found no weaknesses in the protocol.

Goals G10, G15, G21 and G25 are verified as true, leading to the fact that the authentication of principal mobile user Ms and foreign agent B under the supervision of home agent C is successful. This represents the mutual authentication of mobile user and foreign agent.

The use of hash functions throughout the protocol assures the integrity of messages being sent and hence, protects the session key from unauthorized access.

Also, the use of pseudo-identities helps in preserving the privacy of users, thereby providing the feature of user anonymity. Pseudo identities are generated by home agent and sent to mobile user during the registration phase. Therefore, only home agent knows the real identity of mobile user.

The home agent can trace back to the mobile user if there are any problems or issues as only home agent knows the real identity of mobile user. This feature is called traceability.



```
Verification Result
1. Assumptions
2. Protocol Steps
3. Protocol Verification
⊟ Protocol is Verified is True
   ⊞ (0001): C possess at[1] idM is True
   ⊞ (0002): Ms possess at[2] pid is True
   ⊞ (0003): Ms possess at[2] Kem is True
   ⊞ (0004): Ms possess at[2] seq is True
   ⊞ (0005): Ms possess at[2] H((idM,Nh)) is True
   ⊞ (0006): B possess at[3] H((((idM,F((H((idM,Nh)),H((idM,SW))))),Nm),seq)) is True
   ⊞ (0007): B possess at[3] F((H((idM,H((idM,Nh)))),Nm)) is True
   ⊞ (0008): B possess at[3] seq is True
   ⊞ (0009): B possess at[3] IDh is True
   ⊞ (0010): B know at[3] Ms send at[3] (H((((idM,F((H((idM,Nh)),H((idM,SW))))),Nm),seq)),F((H((idM,H((idM,Nh)))),Nm))) is True
   ⊞ (0011): C possess at[4] H((((idM,H((idM,Nh))),Nm),seq)) is True
   ⊞ (0012): C possess at[4] ((F((H((idM,H((idM,Nh)))),Nm)),seq),IDf) is True
   ⊞ (0013): C possess at[4] H(((((((H((((idM,H((idM,Nh))),Nm),seq)),F((H((idM,H((idM,Nh)))),Nm))),seq),IDh),F((H(Kfh),Nf))),Kfh),Nf)) is True
   ⊞ (0014): C possess at[4] F((H(Kfh),Nf)) is True
   ⊞ (0015): C know at[4] B send at[4] H(((((((H((((idM,H((idM,Nh))),Nm),seq)),F((H((idM,H((idM,Nh)))),Nm))),seq),IDh),F((H(Kfh),Nf))),Kfh),Nf)) is True
   ⊞ (0016): B possess at[5] F((H(((H((idM,Nh)),idM),seq)),Nf)) is True
   ⊞ (0017): B possess at[5] F((H((Kfh,Nf)),Nm)) is True
   ⊞ (0018): B possess at[5] F((H((F((H((Kfh,Nf)),Nm)),Nf)),Kfh)) is True
   ⊞ (0019): B possess at[5] F((H(((F((H(((H((idM,Nh)),idM),seq)),Nf)),Nm),F((H(((H((idM,Nh)),idM),Nm)),seq)))),H((idM,Nh)))) is True
   ⊞ (0020): B possess at[5] F((H(((H((idM,Nh)),idM),Nm)),seq)) is True
   ⊞ (0021): B know at[5] C send at[5] F((H(((F((H(((H((idM,Nh)),idM),seq)),Nf)),Nm),F((H(((H((idM,Nh)),idM),Nm)),seq)))),H((idM,Nh)))) is True
   ⊞ (0022): Ms possess at[6] F((H(((H((idM,Nh)),idM),seq)),Nf)) is True
   ⊞ (0023): Ms possess at[6] F((H(((F((H(((H((idM,Nh)),idM),seq)),Nf)),Nm),F((H(((H((idM,Nh)),idM),Nm)),seq)))),H((idM,Nh)))) is True
   ⊞ (0024): Ms possess at[6] F((H(((H((idM,Nh)),idM),Nm)),seq)) is True
   ⊞ (0025): Ms know at[6] B send at[6] ((F((H(((H((idM,Nh)),idM),seq)),Nf)),F((H(((F((H(((H((idM,Nh)),idM),seq)),Nf)),Nm),F((H(((H((idM,Nh)),idM),Nm)),
```

Figure 21

```
seq)))),H((idM,Nh))))),F((H(((H((idM,Nh)),idM),Nm)),seq))) is True
```

Figure 21: Verification result of MAKA scheme in CDVT

Chapter 9: Action plan

Figure 23 represents the Gantt chart for Autumn and Spring semesters and figure 24 represents the Gantt chart for the Summer semester.

The Gantt charts comprise of the following major tasks:

Autumn Semester:

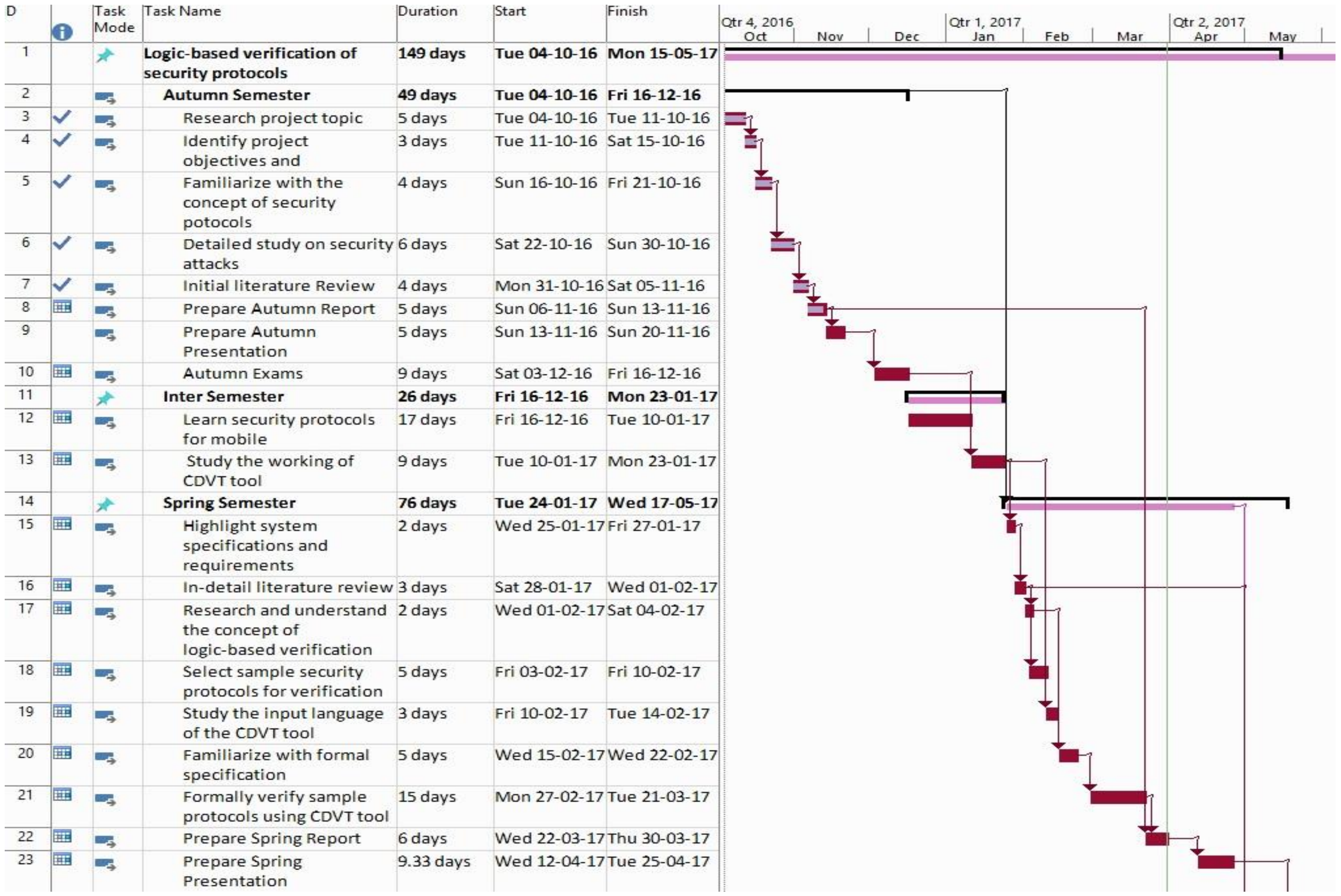
- Research project topic
- Identify project objects and requirements
- Familiarize with the concept of security protocols
- Detailed study on security attacks
- Initial literature review

Spring Semester:

- Study the working of CDVT tool
- In-detail literature review
- Research and understand the concept of logic-based verification
- Study the input language of the CDVT tool
- Familiarize with the formal specification

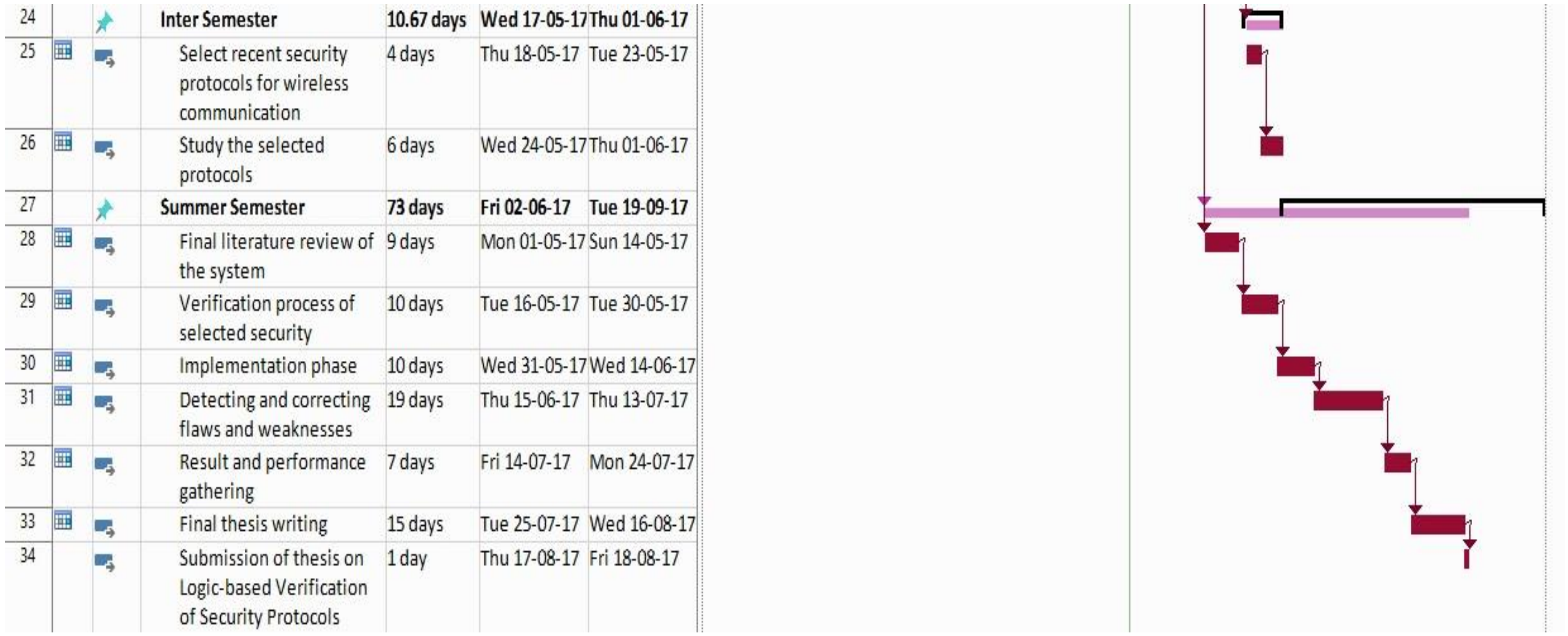
Summer Semester:

- Select eight security protocols for mobile communication for verification
- Research on the selected eight protocols.
- Produce formalized specification of protocols for mobile communication
- Verification process of selected security protocols
- Detecting flaws and weaknesses
- Final thesis submission



D		Task Mode	Task Name	Duration	Start	Finish
1			**Logic-based verification of security protocols**	**149 days**	**Tue 04-10-16**	**Mon 15-05-17**
2			**Autumn Semester**	**49 days**	**Tue 04-10-16**	**Fri 16-12-16**
3	✓		Research project topic	5 days	Tue 04-10-16	Tue 11-10-16
4	✓		Identify project objectives and	3 days	Tue 11-10-16	Sat 15-10-16
5	✓		Familiarize with the concept of security potocols	4 days	Sun 16-10-16	Fri 21-10-16
6	✓		Detailed study on security attacks	6 days	Sat 22-10-16	Sun 30-10-16
7	✓		Initial literature Review	4 days	Mon 31-10-16	Sat 05-11-16
8	▦		Prepare Autumn Report	5 days	Sun 06-11-16	Sun 13-11-16
9			Prepare Autumn Presentation	5 days	Sun 13-11-16	Sun 20-11-16
10	▦		Autumn Exams	9 days	Sat 03-12-16	Fri 16-12-16
11			**Inter Semester**	**26 days**	**Fri 16-12-16**	**Mon 23-01-17**
12	▦		Learn security protocols for mobile	17 days	Fri 16-12-16	Tue 10-01-17
13	▦		Study the working of CDVT tool	9 days	Tue 10-01-17	Mon 23-01-17
14			**Spring Semester**	**76 days**	**Tue 24-01-17**	**Wed 17-05-17**
15	▦		Highlight system specifications and requirements	2 days	Wed 25-01-17	Fri 27-01-17
16	▦		In-detail literature review	3 days	Sat 28-01-17	Wed 01-02-17
17	▦		Research and understand the concept of logic-based verification	2 days	Wed 01-02-17	Sat 04-02-17
18	▦		Select sample security protocols for verification	5 days	Fri 03-02-17	Fri 10-02-17
19	▦		Study the input language of the CDVT tool	3 days	Fri 10-02-17	Tue 14-02-17
20	▦		Familiarize with formal specification	5 days	Wed 15-02-17	Wed 22-02-17
21	▦		Formally verify sample protocols using CDVT tool	15 days	Mon 27-02-17	Tue 21-03-17
22	▦		Prepare Spring Report	6 days	Wed 22-03-17	Thu 30-03-17
23	▦		Prepare Spring Presentation	9.33 days	Wed 12-04-17	Tue 25-04-17

Figure 22: Gantt Chart for Autumn and Spring



24			**Inter Semester**	**10.67 days**	**Wed 17-05-17**	**Thu 01-06-17**
25			Select recent security protocols for wireless communication	4 days	Thu 18-05-17	Tue 23-05-17
26			Study the selected protocols	6 days	Wed 24-05-17	Thu 01-06-17
27			**Summer Semester**	**73 days**	**Fri 02-06-17**	**Tue 19-09-17**
28			Final literature review of the system	9 days	Mon 01-05-17	Sun 14-05-17
29			Verification process of selected security	10 days	Tue 16-05-17	Tue 30-05-17
30			Implementation phase	10 days	Wed 31-05-17	Wed 14-06-17
31			Detecting and correcting flaws and weaknesses	19 days	Thu 15-06-17	Thu 13-07-17
32			Result and performance gathering	7 days	Fri 14-07-17	Mon 24-07-17
33			Final thesis writing	15 days	Tue 25-07-17	Wed 16-08-17
34			Submission of thesis on Logic-based Verification of Security Protocols	1 day	Thu 17-08-17	Fri 18-08-17

Figure 23: Gantt Chart for Summer

Chapter 10: Conclusion

This thesis presents a detailed study on security protocols, formal verification of security protocol, and the Cryptographic protocol Design and Verification tool (CDVT). However, the major focus of this project is on the formal verification of eight recent security protocols for mobile communication and their verification results computed using the CDVT tool. The procedure of formal verification using CDVT has been explained using sample protocols. Security protocols with extensive mathematical calculations and design weaknesses in hardware or some other aspect are sometimes difficult to specify in the CDVT tool, however it does not affect the results obtained from the tool.

Among the eight security protocols for wireless communication, five protocols have been verified to be True by the CDVT tool, and the other three have been verified as False, meaning these are vulnerable to one or more types of attacks. The CRSSP protocol [12], the HIP-based secure SDMN [13], Secure Mobile Wallet [14], P2P service security protocol for M2M communication [16] and the MAKA scheme for GLOMONETs [19] have been verified to be true.

The following are the protocols that have been verified as False:

- Secure patient-health monitoring WBAN [15]
- Communication protocol for vehicular networks [17]
- TTP based High-Efficient Multi-Key Exchange protocol (THMEP) [18]

The patient-health monitoring WBAN [15] failed as the security of the data sink itself is a concern and results in an adversary obtaining information from the data sink. Communication protocol for vehicular networks [17] failed because the protocol had the risk of nodes being compromised as the freshness of message sent could not be proven. THMEP protocol [18] failed as it could not protect against password-guessing attacks which lead the protocol to being vulnerable to other attacks like replay and MITM attacks. More detailed descriptions of these weaknesses are mentioned in the verification results section of each protocol and the objectives of this thesis have been achieved.

References

[1] M. Burrows, M. Abadi and R. and Needham, "A logic of Authentication," *ACM Transactions on Computer Systems (TOCS),* vol. 8, no. 1, pp. 18-36, 1990.

[2] R. Dojen and T. Coffey, "The concept of layered proving trees and its application to the automation of security protocol verification," *ACM Transactions on Information and System Security,* vol. 8, no. 3, pp. 287-311, 2005.

[3] R. Dojen, I. Lasc and T. Coffey, "Establishing and fixing a freshness flaw in a key-distribution and authentication protocol," *Proceedings - 2008 IEEE 4th International Conference on Intelligent Computer Communication and Processing, ICCP 2008,* pp. 185-192, 2008.

[4] M. Ventuneac, R. Dojen and T. and Coffey, "Automated verification of wireless security protocols using layered proving trees," *WSEAS Transactions on Communications,* vol. 5, no. 2, pp. 252-258, 2006.

[5] R. Needham and M. Schroeder, "Using encryption for authentication in large networks of computers," *Communications of the ACM,* vol. 21, no. 12, p. 993–999, 1978.

[6] Microsoft, "Exchanging Session Keys (Windows CE 5.0)," 2017. [Online].

[7] C Boyd and A Mathuria, "Protocols for authentication and key establishment," Berlin, London, Springer-Verlag, 2003.

[8] G. Lowe, "An attack on the Needham-Schroeder public key authentication protocol," *Information Processing Letters,* vol. 56, no. 3, pp. 131-136, November 1995..

[9] G. Lowe, "Some new attacks upon security protocols," *IEEE Computer Society Press, Proceedings of Computer Security Foundations Workshop VIII,* 1996.

[10] J. Nam, S. Kim, S. Park and D. Won, "Security analysis of a nonce-based user authentication scheme using smart cards," *IEICE Transactions Fundamentals,* vol. 90, no. 1, pp. 299-302, 2007.

[11] T. Newe and T. Coffey, "Security Protocols for 2G and 3G Wireless Communications," in *Proceedings of the 1st Intenational Symposium on Information and Communication Technologies,* Dublin, Ireland, September 2003.

[12] S. Gajbhiye, S. Karmakar, M. Sharma and S. Sharma, "Design and analysis of pairing protocol for bluetooth enabled devices using R-LWE Lattice-based cryptography," *Journal of Information Security and Applications* , no. 35, p. 44–50, 2017.

[13] M. Liyanage, B. An, D. J. Anca, Y. Mika and G. Andrei, "Secure communication channel architecture for Software Defined Mobile Networks," *Computer Networks,* no. 114, pp. 32-50, 2017.

[14] Q. Zhen, S. Jianfei, W. Abubaker, Z. Wentao, H. Xiongb and Q. Zhiguang, "A secure and privacy-preserving mobile wallet with outsourced verification in cloud computing," *Computer Standards & Interfaces,* vol. 54, p. 55–60, 2017.

[15] H. Chunqiang, L. Hongjuan, H. Yan, X. Tao and L. Xiaofeng, "Secure and Efficient Data Communication Protocol for Wireless Body Area Networks," *IEEE TRANSACTIONS ON MULTI-SCALE COMPUTING SYSTEMS,* vol. 2, no. 2, pp. 94 - 107, 2016.

[16] W.-S. Bae, "Designing and verifying a P2P service security protocol in M2M environment," *Peer-to-Peer Netw. Appl.,* vol. 9, p. 539–545, 2016.

[17] K. R. Sekhar, T. S. R. Chandra, S. Pooja and S. Tapaswi, "Light weight security protocol for communications in vehicular networks," *Wireless Networks,* vol. 22, p. 1343–1353, 2016.

[18] K.-L. Tsai, Y.-L. Huang, F.-Y. Leu and I. You, "TTP Based High-Efficient Multi-Key Exchange Protocol," *IEEE Access,* vol. 4, pp. 6261 - 6271, 2016.

[19] G. Prosanta and H. Tzonelih, "Lightweight and Energy-Efficient Mutual Authentication and Key Agreement Scheme With User Anonymity for Secure Communication in Global Mobility Networks," *IEEE Systems Journal,* vol. 10, no. 4, pp. 1370 - 1379, 2015.

[20] J. Anca, C. Tom and D. Reiner, "On the prevention and detection of replay attacks using a logic-based verification tool," in *International Conference on Computer Networks,* 2014.

[21] J. Anca, C. Tom and D. Reiner, "A Novel Security Protocol Attack Detection Logic with Unique Fault Discovery Capability for Freshness Attacks and Interleaving Session Attacks," *IEEE Transactions on Dependable and Secure Computing,* 2017.

[22] J. Anca, C. Tom and D. Reiner, "Design guidelines for security protocols to prevent replay & parallel session attacks," *Computers & Security,* vol. 45, pp. 255-273, 2014.

[23] T. Coffey, R. Dojen and T. Flanagan, "Formal verification: An imperative step in the design of security protocols," *Computer Networks,* vol. 43, no. 5, pp. 601-618, 2003.

[24] A. Jurcut, T. Coffey and R. Dojen, "Establishing and fixing security protocols weaknesses using a logic-based verification tool," *Journal of Communications,* vol. 8, no. 11, pp. 795-805, 2013.

[25] T. Coffey, R. Dojen and T. Flanagan, "On different approaches to establish the security of cryptographic protocols," *Proceedings of the International Conference on Security and Management,* pp. 637-643, 2003.

[26] P. Syverson and C. Meadows, "Formal requirements for key distribution protocols," *Proceedings of Advances in Cryptology—EUROCRYPT'94,* p. 320–331, May 1995.

[27] A. W. Roscoe, "Modelling and verifying key-exchange protocols using CSP and FDR," *Proceedings of 8th IEEE Computer Security Foundations Workshop,* p. 98, 1995.

[28] A. Huima, "Efficient infinite-state analysis of security protocols," *Proceedings of FLOC'99 Workshop on Formal Methods and Security Protocols,* July 1999.

[29] S. Gurgens and C. Rudolph, "Security analysis of (Un-)fair Non-repudiation protocols," *Formal Aspects of Security, Lecture Notes in Computer Science 2629,* p. 97–114, 2002.

[30] V. Kessler and G. Wendel, "AUTLOG—Anadvanced logic of authentication," *Proceedings of 7th IEEE Computer Security Foundations,* pp. 90-99, August 1994.

[31] S. Brackin, "Automatically detecting most vulnerabilities in Cryptographic protocols," *DARPA Information Survivability Conference and Exposition Vol.1,* pp. 222-236, 2000.

[32] T. Coffey, R. Dojen and T. Flanagan, "On different approaches to establish the security of Cryptographic Protocols," *Proceedings of SAM'03 (Conference on Security and Management),* vol. 2, pp. 637-643, June 2003.

[33] T. Newe and T. Coffey, "Formal Verification logic for hybrid security protocols," *International Journal of Computer Systems Science & Engineering,* vol. 18, no. 1, pp. 17-25, 2003.

[34] L. C. Paulson, "The inductive approach to verifying cryptographic protocols," *Journal of Computer Security,* vol. 6, no. 1,2, pp. 85-128, 1998.

[35] D. Reiner, C. Jinyong and C. Tom, "On modelling security protocols for logic-based verification," *IET Digital Library,* pp. 79-84, 2014.

[36] C. Hao, J. A. Clark and J. L. Jacob, "Synthesising Efficient and Effective Security Protocols," ARSPA 2004.

[37] T. Coffey and P. Saidha, "A logic for verifying public key cryptographic protocols," *IEE Journal Proceedings - Computers and Digital Techniques,* vol. 144, no. 1, pp. 28-32, 1997.

[38] P. Syverson and P. Van Oorschot, "On unifying some cryptographic protocol logics," *IEEE Symp. on Research in Security and Privacy (Oakland1994 Proceedings),* pp. 14-28, 1994.

[39] Y. Zhang and V. Varadharajan, "A logic for modeling the dynamics of beliefs in cryptographic protocols," *Australian Computer Science Communications* , vol. 23, no. 1, pp. 215-222 , 2001.

[40] R. Dojen and T. Coffey, "A Novel Approach to the Automation of Logic-Based Security Protocol," *WSEAS TRANS. on INFORMATION SCIENCE & APPLICATIONS,* vol. 1, no. 5, pp. 1243-1247, November 2004.

[41] E. Cohen, " First-Order Verification of Cryptographic Protocols," *Journal of Computer Security,* vol. 11, no. 2, pp. 189-216, 2003.

[42] T. Coffey, R. Dojen and T. Flanagan, "On the Automated Implementation of Modal Logics used to verify Security Protocols," *Proceedings of International Symposium on Information and Communication Technologies (Invited Workshop on Network Security and Management),* pp. 324-347 , September 2003.

[43] J. Schumann, "PIL/SETHEO: A Tool for the Automatic Analysis of Authentication Protocols," *Proceedings of International Conference on Computer Aided Verification*, pp. 500-504, July 1999.

[44] E. Saul and A. Hutchison, "SPEAR II: The Security Protocol Engineering and Analysis Resource," *Proceedings of South African Telecommunications, Networks and Applications,* pp. 171-177, September 1999.

[45] R. Hauser and E. Lee, "Verification and Modelling of Authentication Protocols," *Proceedings of Second European Symposium on Research in Computer Security,* pp. 141-154, November 1992.

[46] J. A, C. Tom and D. Reiner, "Establishing and fixing security protocols weaknesses using a logic-based verification tool," *Journal of Communication,* vol. 8, no. 11, pp. 795-806, 2013.

[47] "CDVT Tool User Guide".

[48] I.-L. Kao and R. Chow, "An Efficient and Secure Authentication Protocol Using Uncertified Keys," *Operating Systems Review,* vol. 29, no. 3, pp. 14-21, 1995.

[49] Beller M. J., Chang L.-F. and Yacobi Y, "Privacy and authentication on a portable communications system," *IEEE Journal on Selected Areas in Communications* , vol. 11, no. 2, pp. 821-829, 1993.

[50] R. Dojen and T. Coffey, "A case study on automation of verification logic," *INES'05: IEEE 9th International Conference on Intelligent Engineering Systems - Proceedings*, pp. 139-144, 2005.

[51] R. Dojen, T. Coffey and L. Tian, "A security protocol specification language extension for modal logic-based verification," *IET Conference Publications,* pp. 295-302, 2007.

[52] B. Blanchet, "Security protocol verification: Symbolic and computational models," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*, 2012, pp. 3-29.

[53] T. Newe and T. Coffey, "Formal Testing & Algebraic Modelling Techniques for Verifying Cryptographic Protocols," *1999-EMES-Romania,* 1999.

[54] S. Brackin, "Automatically Detecting Most Vulnerabilities in Cryptographic Protocols," *Proceedings of DARPA Information Survivability Conference and Exposition*, pp. 222-236, January 2000.

[55] Carlsen U, "Optimal privacy and authentication on a portable communications system," *ACM Operating Systems Review 28*, pp. 16-23, 1994.

Appendix A - Sample Protocol Specifications

1. Andrew Secure RPC

```
// Andrew Secure RPC Formal Specification
// ***************************************
// Author: Shireen Kudukkil Manchingal
// Date: 08 Feb 2017

// Protocol specification
// ------------------------
// A, B :  principal
// Kab, Kab1 :  symkey
// Na, Nb, Nb1 :  nonce
// succ :  nonce -> nonce

// 1.     A -> B    :  A, {Na}Kab
// 2.     B -> A    :  {succNa, Nb}Kab
// 3.     A -> B    :  {succNb}Kab
// 4.     B -> A    :  {Kab1, Nb1}Kab


// 1.Initial Assupmtions
// -----------------------
// a.Express A's possessions/knowledge at time t0
A1: A possess at[0] Kab;
A2: A possess at[0] Na;
A3: A know at[0] B possess at[0] Kab;
A4: A know at[0] NOT(Zero possess at[0] Na);

// b.Express B's possessions/knowledge at time t0
A5: B possess at[0] Kab;
A6: B possess at[0] Nb;
A7: B possess at[0] Nb1;
A8: B know at[0] A possess at[0] Kab;
A9: B know at[0] NOT(Zero possess at[0] Nb);
A10: B know at[0] NOT(Zero possess at[0] Nb1);
A11: B possess at[3] Kab1;
```

```
//
****************************************************************
// Protocol Steps
// ---------------
S1: B receivefrom A at[1] A,{Na}Kab;
S2: A receivefrom B at[2] {(F(Na), Nb)}Kab;
S3: B receivefrom A at[3] {F(Nb)}Kab;
S4: A receivefrom B at[4] {(Kab1, Nb1)}Kab;


//
****************************************************************
// Protocol Goals
// ---------------
// a. Mutual Authentication of A and B
G1: A know at[2] B send at[2] {(F(Na),Nb)}Kab;
G2: B know at[3] A send at[3] {F(Nb)}Kab;

// b. Sharing of new key Kab1 between A and B
G3: A possess at[4] Kab1;
G4: B possess at[4] Kab1;
G5: A know at[4] B possess at[4] Kab1;
G6: B know at[4] A possess at[4] Kab1;
G7: A know at[4] NOT( Zero send at[0] {(Kab1, Nb1)}Kab);
```

2. Needham-Schroeder Public Key

```
// Needham-Schroeder Public Key Formal Specification
// ****************************************
// Author: Shireen Kudukkil Manchingal
// Date: 19 Feb 2017


// Protocol specification
// ------------------------
// A,B,S : Principal
// Na,Nb : Nonce
// KPa,KPb,KPs,KSa,KSb,KSs : Key
// KPa,KSa : is a key pair
// KPb,KSb : is a key pair
// KPs,KSs : is a key pair

// 1. A -> S : A,B
// 2. S   -> A : {KPb, B}KSs
// 3. A   -> B : {Na, A}KPb
// 4. B   -> S : B,A
// 5. S   -> B : {KPa, A}KSs
// 6. B   -> A : {Na, Nb}KPa
// 7. A   -> B : {Nb}KPb


// 1.Initial Assupmtions
// -----------------------
// a.Express A's possessions/knowledge at time t0
A1: A possess at[0] KaPriv;
A2: A possess at[0] Na;
A3: A possess at[0] KttpPub;
//A4: A know at[0] B possess at[0] KaPub;
A4: A know at[0] TTP possess at[0] KttpPriv;
A5: A know at[0] NOT(Zero possess at[0] Na);

// b.Express B's possessions/knowledge at time t0
A6: B possess at[0] KbPriv;
A7: B possess at[0] Nb;
A8: B possess at[0] KttpPub;
//A9: B know at[0] A possess at[0] KbPub;
A9: B know at[0] TTP possess at[0] KttpPriv;
A10: B know at[0] NOT(Zero possess at[0] Nb);


//
***************************************************************
// Protocol Steps
// ---------------
S1: TTP receivefrom A at[1] A,TTP;
S2: A receivefrom TTP at[2] {KbPub,B}KttpPriv;
S3: B receivefrom A at[3] {Na,A}KbPub;
S4: TTP receivefrom B at[4] B,A;
S5: B receivefrom TTP at[5] {KaPub,A}KttpPriv;
S6: A receivefrom B at[6] {Na,Nb}KaPub;
```

```
S7: B receivefrom A at[7] {Nb}KbPub;


//
***************************************************************
// Protocol Goals
// ---------------
G1: A know at[6] B send at[6] {Na,Nb}KaPub;
G2: A know at[6] not(Zero send at[0] {Na,Nb}KaPub);
G3: B know at[7] A send at[7] {Nb}KbPub;
G4: B know at[7] not(Zero send at[0] {Nb}KbPub);
G5: A possess at[3] KbPub;
G6: B possess at[6] KaPub;
G7: A know at [2] TTP send at[2] {KbPub,B}KttpPriv;
G8: B know at [5] TTP send at[5] {KaPub,A}KttpPriv;

// Certificate structure: {A,kaPub,TS,TTP}kTTPPriv
```

3. BCY protocol

```
// BCY protocol Formal Specification
// ****************************************
// Author: Shireen Kudukkil Manchingal
// Date: 17 Feb 2017


// Protocol specification
// -------------------------
// U : An identifier of the user
// V : An identifier of the service provider
// S : An identifier of the certification authority
// Kvd+ : V's public key for Diffie-Hellman key agreement
// Kvm+ : V's public key for MSR encryption
// KK : A key-encrypting key
// SK : A session key
// rU : A random nonce generated by U

// 1. V -> U : {V, Kvd+,Kvm+}Ks-
//   U computes Y = {rU}Kvm+, KK={Kvd+}Ku-, SK = {rU}KK
// 2. U -> V : Y, {{U,Ku+}Ks-}rU
//   V computes rU ={Y}Kvm-, KK={Ku+}Kvd-, SK = {rU}KK
// 3. V -> U : {dataV}SK
// 4. U -> V : {dataU}SK

// SKu = {Nu}{Kv2Pub}KuPriv
// SKv = {Nu}{KuPub}Kv2Priv


// 1.Initial Assupmtions
// -----------------------
// a.Express U's possessions/knowledge at time t0
A1: U possess at[0] KuPriv;
A2: U possess at[0] KuPub;
A3: U possess at[0] Nu;
A4: U possess at[0] dataU;
A5: U possess at[0] KttpPub;
A6: U know at[0] (Zero possess at[0] Nu);
A7: U know at[0] (TTP possess at[0] KttpPriv);
A8:  U know at[3] ((U possess at[3] {Nu,Nv}{Kv1Pub}KuPriv AND U
know at[3] V possess at[3] Kv1Priv) IMPLY V possess at[3]
{Nu,Nv}{Kv1Pub}KuPriv);


// b.Express V's possessions/knowledge at time t0
A9: V possess at[0] Kv1Priv;
A10: V possess at[0] Kv2Priv;
A11: V possess at[0] Kv1Pub;
A12: V possess at[0] Kv2Pub;
A13: V possess at[0] dataV;
A14: V possess at[0] KttpPub;
A15: V know at[0] (TTP possess at[0] KttpPriv);
A16: V know at[4] ((V possess at[4] {Nu,Nv}{KuPub}Kv1Priv AND V
know at[4] U possess at[4] KuPriv) IMPLY U possess at[4]
{Nu,Nv}{KuPub}Kv1Priv);
```

```
//
***************************************************************
// Protocol Steps
// ---------------
S1: U receivefrom V at[1] {V,Kv1Pub,Kv2Pub}KttpPriv;
S2: V receivefrom U at[2] {Nu}Kv2Pub,{{U,KuPub}KttpPriv}Nu;
S3: U receivefrom V at[3] {dataV}({Nu}({Kv1Pub}KuPriv));
S4: V receivefrom U at[4] {dataU}{Nu}{KuPub}Kv1Priv;


//
***************************************************************
// Protocol Goals
// ---------------
G1:  U know at[1] V possess at[1] Kv1Priv;
G2:  U know at[1] V possess at[1] Kv2Priv;
G3:  U possess at[1] Kv1Pub,Kv2Pub;
G4:  U possess at[1] {Kv1Pub}KuPriv;
G5:  U possess at[1] {Nu}{Kv1Pub}KuPriv;
G6:  U know at[1] (not(Zero possess at[0] {Nu}{Kv1Pub}KuPriv));
G7:  V know at[2] U possess at[2] KuPriv;
G8:  V possess at[2] KuPub;
G9:  V possess at[2] {KuPub}Kv1Priv;
G10: V possess at[2] {Nu}{KuPub}Kv1Priv;
G11: V know at[2] (not(Zero possess at[0] {Nu}{KuPub}Kv1Priv));
G12: U know at[3] not(Zero send at[0] {dataV}{Nu}{Kv1Pub}KuPriv);
G13: U know at[3] V send at[3] {dataV}{Nu}{Kv1Pub}KuPriv;
G14: V know at[4] not(Zero send at[0] {dataU}{Nu}{KuPub}Kv1Priv);
G15: V know at[4] U send at[4] {dataU}{Nu}{KuPub}Kv1Priv;
```