Development of Security Strategies using Kerberos in Wireless Networks

Yöney Kırsal Ever

School of Engineering and Information Sciences

Middlesex University

A thesis submitted to Middlesex University in fulfillment of the requirements for degree of Doctor of Philosophy

January 2011

Dedicated to,
My parents, Hamide and Ayhan KIRSAL
My love, Dr. Enver EVER

Abstract

Authentication is the primary function used to reduce the risk of illegitimate access to IT services of any organisation. Kerberos is a widely used authentication protocol for authentication and access control mechanisms.

This thesis presents the development of security strategies using Kerberos authentication protocol in wireless networks, Kerberos-Key Exchange protocol, Kerberos with timed-delay, Kerberos with timed-delay and delayed decryption, Kerberos with timed-delay, delayed decryption and password encryption properties. This thesis also includes a number of other research works such as, frequently key renewal under pseudo-secure conditions and shut down of the authentication server to external access temporarily to allow for secure key exchange.

A general approach for the analysis and verification of authentication properties as well as Kerberos authentication protocol are presented. Existing authentication mechanisms coupled with strong encryption techniques are considered, investigated and analysed in detail. IEEE 802.1x standard, IEEE 802.11 wireless communication networks are also considered. First, existing security and authentication approaches for Kerberos authentication protocol are critically analysed with the discussions on merits and weaknesses. Then relevant terminology is defined and explained.

Since Kerberos exhibits some vulnerabilities, the existing solutions have not treated the possibilities of more than one authentication server in a strict sense. A three way authentication mechanism addresses possible solution to this problem. An authentication protocol has been developed to improve the three way authentication mechanism for Kerberos. Dynamically renewing keys under pseudo-secure situations involves a temporary interruption to link/server access. After describing and analysing a protocol to achieve improved security for authentication, an analytical method is used to evaluate the cost in terms of the degradation of system performability. Various results are presented.

An approach that involves a new authentication protocol is proposed. This new approach combines delaying decryption with timed authentication by using passwords and session keys for authentication purposes, and frequent key renewal under secure conditions. The analysis and verification of authentication properties and results of the designed protocol are presented and discussed.

Protocols often fail when they are analysed critically. Formal approaches have emerged to analyse protocol failures. Abstract languages are designed especially for the description of communication patterns. A notion of rank functions is introduced for analysing purposes as well. An application of this formal approach to a newly designed authentication protocol that combines delaying the decryption process with timed authentication is presented.

Formal methods for verifying cryptographic protocols are created to assist in ensuring that authentication protocols meet their specifications. Model checking techniques such as Communicating Sequential Processes (CSP) and Failure Divergence Refinement (FDR) checker, are widely acknowledged for effectively and efficiently revealing flaws in protocols faster than most other contemporaries. Essentially, model checking involves a detailed search of all the states reachable by the components of a protocol model. In the models that describe authentication protocols, the components, regarded as processes, are the principals including intruder (attacker) and parameters for authentication such as keys, nonces, tickets, and certificates. In this research, an automated generation tool, CASPER is used to produce CSP descriptions. Proposed protocol models rely on trusted third parties in authentication transactions while intruder capabilities are based on possible inductions and deductions. This research attempts to combine the two methods in model checking in order to realise an abstract description of intruder with enhanced capabilities. A target protocol of interest is that of Kerberos authentication protocol.

The process of increasing the strength of security mechanisms usually impacts on performance thresholds. In recognition of this fact, the research adopts an analytical method known as spectral expansion to ascertain the level of impact, and which resulting protocol amendments will have on performance. Spectral expansion is based on state exploration. This implies that it is subject, as model checking, to the state explosion problem. The performance characteristics of amended protocols are examined relative to the existing protocols.

Numerical solutions are presented for all models developed.

Acknowledgements

I gratefully thank my director of studies Dr. Orhan Gemikonakli for introducing me to this area of research and for his outstanding supervision, throughout this research. Dr. Gemikonakli has contributed in every stage of my PhD studies. The technical discussions and guidance have been highly motivating, productive, and have given me a great deal of professional satisfaction. This work would not be possible without his precious contributions.

Special thanks due to my second supervisor Dr. Shahedur Rahman for his important supervision. The support he provided was very useful and helped me understand many concepts. I am thankful to his support and guidance.

I am also grateful to study at Middlesex University and many thanks to my friends in the Computer Communications Department, for their optimism and support.

Also, I would like to send the most sincere thanks to my mum, Hamide, my dad, Ayhan and my brother, Yönal for their support.

Finally, words are not enough for him, his love and support. I am very lucky to meet Enver. During difficult times he was always with me with his altruism.

List of Publications

The work presented in this thesis has given rise to the following publications.

1. Kirsal Y., A. Eneh and O. Gemikonakli (2005), ”A Solution to the Problem of Trusted Third Party for IEEE 802.11b Networks”. PGNET2005, Liverpool UK, pp.333-339.

2. Kirsal Y. and O. Gemikonakli (2006), “An Authentication Protocol to Address the Problem of the Trusted 3rd Party Authentication Protocols”, In Proceedings of Innovative Algorithms and Techniques in Automation, Industrial Electronics and Telecommunications (CISSE 2006), Springer Netherlands, pp. 523-526.

3. Kirsal Y. and O. Gemikonakli (2007), “Frequent Key Renewal under Pseudo-Secure Conditions for Increased Security in Kerberos Authentication and Its Impact on System Performability”, 3rd International Conference on Global E-Security, University of East London (UeL), Docklands, UK.

4. Kirsal Y., (2007), Poster Presentation titled “Development of Security Strategies Using Kerberos in Wireless Networks”, Women in Computing Research London Hopper 2007, BCS London Offices, UK and in The Richard Tapia Celebration of Diversity in Computing Conference 2007, Orlando, Florida, USA.

5. Kirsal Y. and O. Gemikonakli (2007), “Further Improvements to the Kerberos Timed Authentication Protocol”, In Proceedings of Novel Algorithms and Techniques In Telecommunications, Automation and Industrial Electronics (CISSE 2007), Springer Netherlands, pp. 550-554.

6. Kirsal Y. and O. Gemikonakli (2008), “Improving Kerberos Security through the Combined Use of the Timed Authentication Protocol and Frequent Key Renewal”, 6th IEEE International Conference on Cybernetic Systems 2008(CIS’08), Middlesex University, London, UK, pp. 153-158.

7. Kirsal Y. and O. Gemikonakli (2009), “Analysing the Kerberos Using CSP-Rank Functions”, Kaspersky Lab Student Conference on Computer Security Issues, Moscow, Russia.

8. Kirsal Y. and O. Gemikonakli(2009), “Analysing the Kerberos Timed Authentication Protocol Using CSP-Rank Functions”, Global Security, Safety, and Sustainability (ICGS3’09), Springer Berlin Heidelberg, vol.45, pp. 56-63

9. Ever E., Y. Kirsal and O. Gemikonakli(2009), “Performability Modelling of a Kerberos Server with Frequent Key Renewal under Pseudo-Secure Conditions for Increased Security”, IEEE International Conference on the Current Trends in Information Technology (CTIT’09), Higher Colleges of Technology, Dubai, pp. 91 - 96.

10. Kirsal Y., O. Gemikonakli and S. Rahman (2010), “Analysing the Combined Kerberos Timed Authentication Protocol and Frequent Key Renewal Using CSP and Rank Functions”, International Journal of IET Information Security **(submitted)**.

Contents

List of Figures

List of Tables

Acronyms

CSP Communicating Sequential Processes. iii, 3–6, 15, 16, 18, 19, 21, 22, 27, 28, 30, 36, 40, 41, 43–46, 55, 58–60, 64, 81, 82, 86, 90, 91, 93, 97–102

EAP Extensible Authentication Protocol. 10–12, 30, 99

EAPOL EAP Over LANs. 32

FDR Failure Divergence Refinement. iii, 4, 6, 15, 21–23, 28, 30, 35, 36, 40, 41, 50, 51, 54, 63, 79, 81–84, 86, 90, 91, 95, 97, 99–102

IAKERB Initial Authentication and Pass Through Authentication using Kerberos V5 and GSS-API. 12

KDC Key Distribution Centre. 8, 12, 13, 26–28, 31–33, 68, 71, 73, 74, 77, 99, 101, 102

MQL Mean Queue Length. 69, 70, 74, 77, 100

PEPA Performance Evaluation Process Algebra. 26

PKCROSS Public Key Cryptography for Cross-Realm Authentication in Kerberos. 12

PKINIT Public Key Cryptography for Initial Authentication in Kerberos. 12

PKTAPP Public Key Utilising Tickets for Application Servers. 12

RoSSI Random Selection of Sign-on Information. 10

RSN Robust Security Network. 10

TAM Tripartite Authentication Mechanism. 10, 31

TGi IEEE 802.11i Task Group on Security. 3, 10

TGS Ticket Granting Service. 8, 12, 18, 19, 31–33, 99

WEP Wired Equivalent Privacy. 9

WPA Wi-Fi Protected Access. 9, 26

Glossary of Symbols

Logic

Notation	Meaning	Example
$\Box$	end of proof	
$\neg$ P	not P(P is not true)	$\neg$ 3$\geq$5
P $\wedge$ Q	P and Q (both true)	x $\leq$ x+1 $\wedge$ x $\neq$ x+1
P $\Rightarrow$ Q	if P then Q	x < y $\Rightarrow$ x $\leq$ y
P $\vdash$ Q	Q is derivable from P	x $\Rightarrow$ y $\vdash$ $\neg$y $\Rightarrow$ $\neg$x
P $\Leftrightarrow$ Q	P is true if Q is true, P is false if Q is false	$x + 5 = y + 2 \Leftrightarrow x + 3 = y$
P $\triangleq$ Q	P is defined to be logically equivalent to Q	
P $\overset{def}{=}$ Q	P is defined to be logically equivalent to Q	

Processes

Notation	Meaning
a $\Rightarrow$ P | b $\Rightarrow$ Q	*a* then *P* choice *b* then *Q* (where provided a$\neq$b)
P || Q	the *P* in parallel with *Q*
P $\underset{x}{\parallel}$ Q	P and Q are synchronised on all events in *X*, but they are allowed to perform events outside *X* freely
$\Box$ S	general choice
P ||| Q	concurrent runs of process *P* in the set of *Q* (*P* interleave *Q*)
b!e	on channel *b*, output value of *e*
b?x	on channel *b*, input to *x*
l!e?x	call of shared subroutine named *l* with value parameter *e* and results to *x*
P `sat` S	process *P* satisfies specification *S*
tr	an arbitrary trace of the specified process
P $\sqsubseteq$ Q	process Q can do at least as much as process P, if P and Q are comparable elements of some partially ordered set
P $\sqcup$ Q	deterministic P or Q (The least upper bound of two processes P and Q)

Functions

Notation	Meaning
f : x $\hookrightarrow$ y	a function of *f* which maps each member of *A* to a *distinct* member of *B*

Traces

Notation	Meaning	Example
$\langle\ \rangle$	empty trace	
$\langle \mathrm{a} \rangle$	the trace containing only *a*	
$\langle \mathrm{a, b, c} \rangle$	the trace with three elements	
s $\Downarrow$ c	the communications on channel, *c* recorded in s	$\langle \mathrm{c.1,a.4,c.3,d.1} \rangle \Downarrow \mathrm{c} = \langle 1,3 \rangle$
s $\upharpoonright$ A	*s* restricted to *A*	$\langle \mathrm{b,c,d,a} \rangle \upharpoonright \{\mathrm{a,c}\} = \langle \mathrm{c,a} \rangle$

Chapter 1

Introduction

1.1 Introduction

The popularity and the use of computers and network-based electronic devices have increased rapidly. Since computers are employed in all phases of today's world, many large collections of materials, including classified confidential information that attract various illegitimate attention, are available electronically. This makes engineers and scientists aware of the need to protect data, resources and systems against network-based attacks and unauthorised access. Providing privacy and data integrity have become more crucial. In addition to these requirements, availability, which can be defined as the requirement that components and resources are continuously in operative mode, are the main goals of network security.

Apart from the use in everyday lives of people all over the world, computer and communication systems are very widely used in research, industry and business. As the users' demands increase, the complexity of computer networks, network products, communication systems and information systems also increases. As a result of this, it becomes more difficult to understand and handle various components of systems which interact in these complex environments, and to make sure that all the implications have been covered and considered properly.

Despite developments in computer network security, incidences of computer crime, attacks on computer systems resources such as break-ins, have continued to increase in complexity and strategy. For this purpose, attack prevention, authentication, and access control issues are explored widely. The difference between authentication and access control concepts is that authentication deals with identity verification before access is granted and whose result is used as input to all other security provisions within networks, where access control deals with the rights and privileges of users [Kahate, 2009], [Stallings, 2010]. Authentication mechanisms are founded on protocols together with cryptographic algorithms.

When communication networks are implemented without authentication mechanisms, there is a risk of consuming resources and information by committing them to attacks. Also it is possible to end up with a system with inadequate confidentiality, integrity and availability characteristics. In the modern world, with fast improving technology, highly competitive research and industry environment, such a risk is not acceptable.

Cryptographic algorithms are simply mathematical functions that transform plaintexts to ciphertexts, so that only entities that possess appropriate requirements, such as correct keys, can reverse the operation to retrieve plaintexts. On the other hand, protocols are step by step sequences of achieving authentication. The emphasis of this project is on authentication protocols. Authentication protocols govern the establishment of secured communication between principals of a network, within the network. Computer systems and users are considered as principals.

Authentication servers and ticket-granting services are the trusted third parties in process of authentication by issuing communicating principals with necessary authentication tokens whereas the channels are the links between the principals and the servers. The tokens include items such as keys, digital signatures, and digital certificates. Fundamentally, this project investigates the development of protocols that are used for authentication purposes.

1.2 Scope of Investigation

This research aims to develop authentication frameworks for the design and implementation of effective and efficient mechanisms for Kerberos authentication protocol used in wireless communication networks. The design of authentication protocols spans well over three decades with the foundation of authentication protocols by [Needham and Schroeder, 1978]. This work employed the encryption algorithms invented by [Diffie and Hellman, 1976]. Following these, many works have emerged being motivated by evolution of flaws discovered against published protocols [Abadi and Needham, 1996], [Lowe, 1995], [Eneh *et al.*, 2006] and [Shaikh and Bush, 2006]. The target authentication mechanisms are those to be used in wireless communications. Kerberos is widely-deployed system that provides authentication and the establishment of secure channels in open networks [Neuman and Ts'o, 1994]. With the continuous failure of authentication protocols, it becomes imperative that published protocols be verified regularly to be sure that the claimed features exist. This requirement becomes higher for authentication protocols deployed in communication network scenarios where trusted third parties can hardly be realised, and regarded as hostile environments.

While Kerberos approach has been proposed as a standard for enhanced security in IEEE 802.11i Task Group on Security (TGi) [Kâafar *et al.*, 2004], currently there is no valid proposals using a Kerberos-like mechanism to provide authentication in a wireless network, preventing from cryptographic attacks and handling fast and secure handovers. In this research, authentication protocols for the IEEE 802.11 networks, based on the IEEE 802.11i works are proposed and Kerberos protocol is used to provide a framework for these purposes.

Generally, authentication protocols are verified by the use of exact models representing the protocols. Usually, the representative models describe the principals including intruders or attackers. While it may be easier to describe an attacker with simple attributes, it is more difficult to describe attackers with complex attributes in situations where third parties can easily be realised. The current methods of verification adopt formal approaches for code generation and analysis of protocols [Meadows, 1995]. Within the scope of formal methods for verifying cryptographic protocols, techniques based on model checking have become widely accepted [Meadows, 1995], [Schneider, 1998]. Model checking methods are classified as state exploration methods, and have made reasonable progress in evolution and adaptation for effective and efficient verification of protocols [Ryan *et al.*, 2000], [Roscoe, 2005]. In regular system design terms, protocol verification provides the assurance that protocol implementation satisfies requirements specification. Besides model checking, other methods for verifying cryptographic protocols include, among others, those of belief logics, inductive proof systems, cryptographic or provable security [Meadows, 1995], [Ryan *et al.*, 2000]. However, model checking has gained the reputation of being more effective and efficient than other methods. Model checking is a state exploration approach that provides for automatic verification of concurrent systems by performing comprehensive searches of all reachable states of systems [Cremers *et al.*, 2009]. In concurrent systems, states execute together in parallel by describing activities of principals in a communication network [Schneider, 1999], [Cremers and Lafourcade, 2007]. Thus, principals of a communication network are concurrent systems. A popular mathematical algebra namely CSP developed by [Hoare, 1985] has recorded the most remarkable progress in the perspective of describing concurrent systems [Roscoe, 2005].

FDR is a model checking tool developed by Formal Systems, that establishes results about concurrent and reactive systems described in CSP [Ryan *et al.*, 2000]. FDR functions check if implementation refines the specification of the system. In this research, systems considered are authentication protocols. Despite the unrivalled power of description inherent in the CSP language, the process of producing CSP scripts by hand remains difficult, error-prone, and time-consuming. CASPER, a compiler, which translates high level description of protocols to CSP codes by [Lowe, 1997], is progressing and is achieving the required results.

The process of increasing the strength of security mechanisms usually impacts on performance degradation. In recognition of this fact, the research adopts an analytical method known as spectral expansion [Chakka and Mitrani, 1994], [Ever, 2007] to ascertain the level of impact, which resulting protocol amendments will have on performance. In other words, addition of extra security features, in the form of additional encryption and message content, will amount to cost in terms of network performance. Performance and availability analysis predict and detect possible drawbacks of the authentication protocols. In this research, performance and availability, in a composite manner namely, performability is used to be able to abstract the exact behaviour of the communication networks. Analytical models have been developed to evaluate the performability of the proposed approaches.

1.3 Contributions of the Thesis

The contributions of the Thesis are:

1. A Framework for Solution to the Trusted Third Party for Wireless Communication Networks
2. An Authentication Protocol to Address the Delay Decryption Property
3. A New Solution for Frequent Key Renewal under Pseudo-Secure Conditions
    (a) Construction of Rank Functions for Frequent Key Renewal
4. Performability Modelling of a Kerberos Server with:
    (a) Frequent Key Renewal under Pseudo-Secure Conditions
    (b) Server Breakdowns and Repairs
5. Modelling Attacker with Increased Powers by Inductions and Deductions

1.4 Outline of the Thesis

Chapter Two introduces the domain of the research by providing a critical review of relevant literature. Existing authentication protocols as well as Kerberos authentication protocol in wireless communication networks are critically analysed and compared. Security considerations for Kerberos are explained in detail. Existing model checking techniques and formal verification methods for the systems under study are investigated and critically analysed. Also, detailed explanation of automated code generation is given. Studies on performance evaluation of security policies are investigated. Detailed explanation of perfomance evaluation techniques is given in this chapter.

The existing solutions commonly known in literature do not treat the possibilities of more than one authentication server in a strict sense. A three way authentication mechanism is a possible solution for this matter. The design of a new framework is the main focus of Chapter Three. The framework using three-way authentication method (Kerberos) to address possible vulnerabilities is introduced. The practical applicability of this framework is also investigated. Verification of the designed framework is provided by CSP. Verification results are presented, and alongside critical analyses, significant comparisons are made .

A similar approach that is used in Chapter Three, is used for the work presented in Chapter Four. The design of this new protocol depends on *delay decryption* property of Kerberos [Neuman and Ts'o, 1994] within the designed framework in Chapter Three. The design of the protocol, its availability and its verification are the main concerns of this chapter. A new authentication protocol, as well as its entities are defined and introduced. Also, the use of this protocol and its entities are validated.

Chapter Five presents an approach of dynamically renewing keys under pseudo-secure situations, thus significantly reducing the chances of potential intruders. The proposed approach involves secure key distributions at various intervals. Pseudo-secure situations are generated from secure random situations, and that are very hard for an observer to distinguish from true random secure conditions. During key distribution, link/server access is temporarily interrupted. The access restriction happens for short intervals. An approach that involves a new authentication protocol that combines frequent key renewal with *timed* authentication is also modelled. The analysis and verification of authentication properties and results of this modelled protocol are presented and discussed. To find out unassailable attacks of the designed protocol, critical analysis of the protocol and authentication specifications are done through automated code generation. The time required to break the encrypted messages were considered in previous chapters (Chapter Three and Chapter Four). In Chapter Five, these breaking times are compared with the breaking times of modelled and developed protocol of this chapter. Results are presented and discussed. Also, a CSP model and construction of rank functions for the modelled protocols are presented. The CSP model and Rank functions are based on the extensive rules of [Schneider, 1998] and [Roscoe, 2005].

In Chapter Six, performance evaluation techniques are employed for security protocols. Analytical models are developed in order to analyse the effects of frequent key renewal under pseudo-secure conditions of Chapter Five, and to evaluate the cost in terms of the degradation of system performance with server failures. Numerical results are presented and discussed. While key distribution times depend on network characteristics such as size, speed, congestion etc., the intervals between key renewals can be determined by the mean values of decryption times. Unlike the previous studies, the server failures [Brennen, 2004] are also considered together with the interruptions for key distributions. The chapter shows the two dimensional Markov model of a Kerberos server considered and proposes a steady state solution approach. Then, numerical results for the performability measures are presented to show the effects of key renewals as well as server failures.

Formal methods for verifying security properties of cryptographic systems, designed for the purposes of assuring that systems satisfy their respective security requirements, tend towards analysing attacker potentials [Lowe, 1996], [Paulson, 1998], [Roscoe *et al.*, 2009]. The approaches of deductive, inductive or both, are based on making inferences about the attacker with a view to provide security offerings that can resist the activities of attackers. In Chapter Seven, a new attacker model is constructed with inductive and deductive approaches. Attacker definition is built into the CSP models of proposed protocols mentioned in Chapters 3, 4, 5, and subsequently tested using FDR. It is widely acknowledged that model checking offers more detailed analysis and discovers attacks faster [Roscoe, 2005] and [Roscoe *et al.*, 2009]. The inductive approach used in [Paulson, 1998] is criticised for failing to include reception events in its model [Ryan *et al.*, 2000]. Therefore, following is the discussion of the attacker under possible deductions, and then the extension of the approach to include possible inductions.

Chapter Eight summarises the main contributions of the thesis and outlines some possible avenues for future studies.

Chapter 2

Literature Review

2.1 Introduction

Owning to the growing popularity and use of computers and network-based devices, providing privacy and data integrity have become crucial, in order to protect data, resources and systems from attacks and unauthorised access. For purposes of attack prevention, authentication and access control play a vital role [Abadi and Needham, 1996]. In recent years, to meet the increasing demands in secure computer communications, various security protocols have been developed. Most of these protocols agreed upon a cryptographic key or achieved authentication specifications. Needham and Schroeder demonstrated the use of the cryptographic algorithms in the design of a set of cryptographic protocols that revolutionised the paradigm of authentication within the sphere of security mechanisms for networks and distributed systems [Needham and Schroeder, 1978]. In order to meet increasing demands, various security protocols have been developed. Kerberos is one of these commonly used mechanisms [Kohl and Neuman, 1993]. In this chapter existing security and cryptography techniques of Kerberos are critically analysed. Solution methods for this authentication protocol are compared. In addition, wireless communication networks and their security aspects are also critically analysed. Detailed explanations of analyses techniques and tools are given. Performance evaluation techniques for communication networks, performance studies for network security are also analysed to look at performance degradation that caused by new security measures.

2.2 Evaluation of Existing Methods and Solution Techniques for Kerberos

In this section, the existing solution techniques for Kerberos authentication protocol are analysed. Kerberos security considerations and basic operation of Kerberos in wireless communication networks are explained in detail. Additionally, solution techniques of Kerberos for wireless communication networks are elaborated. The merits and weaknesses of Kerberos and its solution techniques are critically analysed.

2.2.1 Kerberos and Security Considerations

Since the publication of protocols by Needham and Schroeder for both the conventional or shared key algorithms and the public key algorithms [Needham and Schroeder, 1978], credible and popular spin-offs have emerged. In the same study, they considered three functions, which include; establishment of authenticated interactive communication, authenticated one-way communication (mailing systems), and signed communication (origin and content integrity authenticated to a third party). The establishment of authenticated interactive communication is of special interest to this research.

Kerberos authentication protocol was designed as part of the project Athena, provides secret key (symmetric key) cryptography for the authentication of client-server applications. As an authentication server, it is based in part on Needham-Schroeder Authentication Protocol [Needham and Schroeder, 1978], but with changes to support the needs of the environment for which it was developed. It uses key distribution. Clients and servers use digital tickets to identify themselves to the network and secret cryptographic keys for secure communications [Neuman and Ts'o, 1994]. Kerberos architecture is divided into two core elements, Key Distribution Centre (KDC) and Ticket Granting Service (TGS) [Kohl and Neuman, 1993], [Neuman and Ts'o, 1994]. The KDC stores authentication information while TGS holds digital tickets for identifying clients and servers of networks. The KDC acts as a trusted third party in performing these authentication services. Kerberos provides a mutual authentication between a client and a server. Mutual authentication of Kerberos uses a technique that requires a password. Many authentication techniques, (such as Wide Mouthed Frog Protocol and Splice/AS Protocol [Schneier, 1996]) send passwords as clear text (i.e. they are not secure), allowing them to be compromised by an unauthorized party. Kerberos solves this problem via encryption. Rather than sending the password, an encrypted key derived from the password is communicated and thus the password is never sent clearly. This technique can be used to authenticate a client and can also be used for mutual authentication of a server. Once authentication takes place, all further traffic is also encrypted, allowing new encryption keys to be communicated securely.

However, as it is the case for commonly known cryptographic protocols (such as Woo-Lam Protocol [Woo and Lam, 1994] and KSL Protocol [Kehne *et al.*, 1992]) in literature [Bellovin and Merritt, 1991], Kerberos is prone to various types of attacks [Geier, 2005]. These vulnerabilities include aided attacks such as the replay of old messages, password guessing, sniffing, jamming, masquerading, injection, cracking and rogue access points, denial of service attacks and session hijacking. The following limitations and issues of Kerberos are considered when implementing an authentication service based on Kerberos [Kohl and Neuman, 1993]:

1. Denial of service attacks: This type of attack is beyond the capabilities of Kerberos, and the detection and solution remain the responsibility of administrators or could be delegated to other applications.

2. Secrecy: Participating principals are required to maintain the secrecy of secret keys. Failure in this situation can allow an intruder to masquerade as a targeted principal.

3. Dictionary attacks: Kerberos provides a weak security for password guessing attacks. Sometimes additional relevant password management procedures are preferred for use.

4. Clock synchronisation: Message times are used to ascertain freshness to avoid replay attacks. There is a need for the principals, (i.e. clients), clocks within the network to be loosely synchronised to enable the servers, in particular, to reliably detect replays.

5. Identifiers recycling: The identifiers of the principals, that leave the network should be removed, in order to avoid the possibility of a new principal inheriting the privileges of another principal that had left earlier. It is remarkable that principal identifiers are not recycled on a short-term basis in Kerberos.

2.2.2 Basic Operation of Kerberos in Wireless Communication Networks

Owing to the growing popularity and use of IEEE 802.11 wireless network communications, there are several different approaches for authentication and encryption of these communication networks such as Wired Equivalent Privacy (WEP) and Wi-Fi Protected Access (WPA). These expanding approaches have fulfilled the security challenges of the technologies. Some of these include, the per-packet authentication approach described by [Mishra and Arbaugh, 2002], affecting access point authentication [Needham and Schroeder, 1978], and the mutual authentication based solution [Schneider, 1998], [Schneider, 1999].

Nonetheless, these approaches inadvertently rely on the notion of explicit trust on the authentication server(s). The IEEE TGi made the initial attempt by developing the Robust Security Network (RSN), which utilises the IEEE 802.1x port-based security standard to provide security services. The IEEE 802.1x standard establishes the Extensible Authentication Protocol (EAP) framework [Mishra and Arbaugh, 2002] and [Vollbrecht *et al.*, 2001]. This EAP framework allows the use of various authentication mechanisms or approaches on top of the EAP layer [Vollbrecht *et al.*, 2001]. It uses three entities as authentication components namely, the supplicant (client), the authenticator (access point) and the authentication server (RADIUS) [Aboba and Calhoun, 2003]. This authentication framework implies that explicit 'trust' is placed on the authentication server, while the access point and wireless clients are suspects [Eneh *et al.*, 2004]. Figure 2.1 adapted from [Mishra and Arbaugh, 2002], shows the layers of the EAP protocol stack. Nevertheless, this study reveals that current RSN architecture only supports mutual authentication between wireless clients and their corresponding access points. Where consideration is given to this scenario, the issues are based on some assumptions of the integrity of the communication channel between the access point and authentication server [Congdon *et al.*, 2003]. Therefore, the consequence is that wireless clients based on 802.11b standard and whose security provision rely on the 802.1x standard, are still liable to session hijacking and masquerading attacks on the link between authenticator and authentication server.

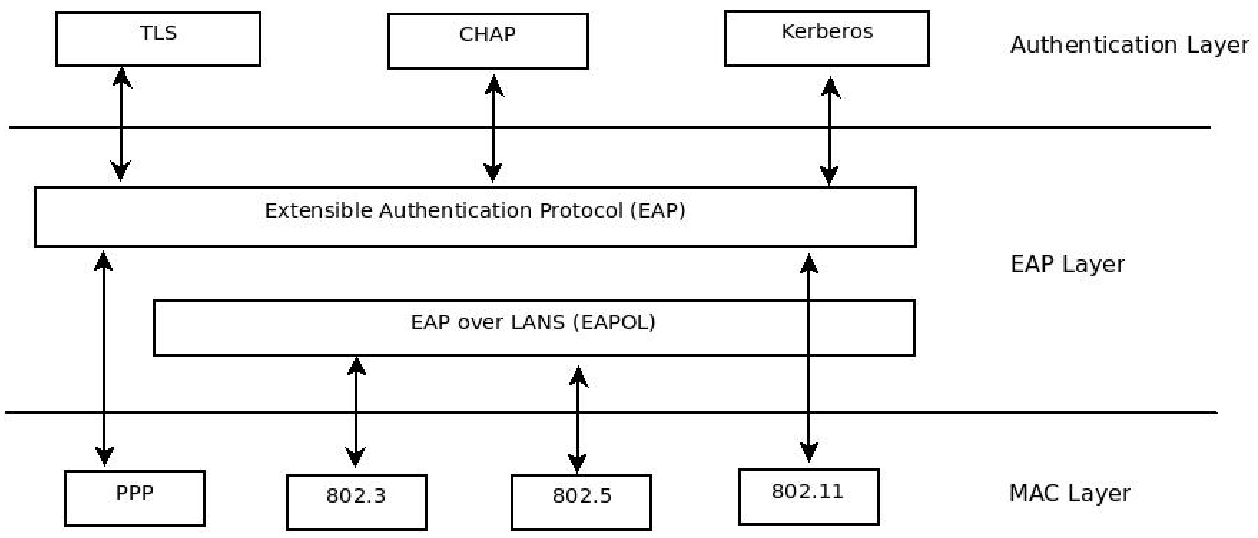


Figure 2.1: The EAP stack

Additionally, a framework, namely Tripartite Authentication Mechanism (TAM) that relies on 802.1x standard is offered [Eneh *et al.*, 2004]. The proposed framework's three entities (supplicant, authenticator, authentication server) mutually authenticate with each other prior to data traffic. It was built on an assumption that none of the parties should be trusted in a wireless LAN communication environment. TAM belongs to EAP authentication layer protocol. It incorporates with a user authentication module that is named as Random Selection of Sign-on Information (RoSSI) functions by making a number of challenges selected at random to the user and include facilities for re-authentication for wireless client stations whose sessions become idle for a period of time. This ensures that an attacker or a rogue wireless workstation steals no valid wireless communication session.

In the study,[Marin-Lopez *et al.*, 2009], propose an architecture aimed for reducing the latency of network access authentication based on the EAP. Their architecture is based on the design of a new EAP method that uses standalone authenticator, and does not require any change to the EAP specification or the specifications of IEEE 802.11. In this study, it is expressed that traditional EAP authentication solutions for seamless mobility require modifications in the current standards and wireless technologies and a new framework is introduced with current standards. It is verified that the architecture does not introduce additional latency which compromises the fast re-authentication process [Marin-Lopez *et al.*, 2009].

In wireless networks, although Kerberos relies on the provisions of IEEE 802.1x standard, owing to the fact that, its operation is system and application independent, security features for authentication are independent as well. Kerberos protocol assumes that initial transactions take place on an open network where clients and servers may not be physically secure and packets travelling on the network can be monitored and even possibly be modified [SECWP, 2007].

Zrelli and Shinoda designed the integration of Kerberos protocol as an authentication method in existing EAP-based authentication frameworks. They defined the architectural elements and their interactions, and the encapsulation of Kerberos messages in EAP packets are specified [Zrelli and Shinoda, 2007]. In the same study, it is expressed that, the use of Kerberos as an EAP authentication mechanism would allow institutions to manage their individuals using a Kerberos system to re-use the same credentials for network access authentication instead of managing a different set of credentials such as Unix passwords or public key certificates. The integration of Kerberos protocol as an authentication method in existing EAP-based authentication frameworks is designed. The architectural elements, their interactions are also defined and the encapsulation of Kerberos messages in EAP packets are specified [Zrelli and Shinoda, 2007].

Due to the critical function of the KDC, multiple KDCs are normally utilised, where each KDC stores a database of users, servers, and secret keys. However, since the KDC stores secret keys for every user and server on the network; they must be kept completely secure. If an attacker were to obtain administrative access to the KDC, the attacker would have access to the complete resources of Kerberos realm.

Kerberos tickets are cached on the client systems. If an attacker gains administrative access to a Kerberos client system, he can impersonate the authenticated users of that system. In other words, the authentication service authenticates the client and replies to the client with a ticket to the TGS. The TGS receives the ticket from the client and checks its validity and replies to the client with a new ticket for the server the client wishes to use. In order to prevent ticket hijacking, Kerberos KDC must be able to verify that the user who is presenting the ticket is the same user to whom the ticket was issued [Neuman and Ts'o, 1994]. This is shown in Figure 2.2.

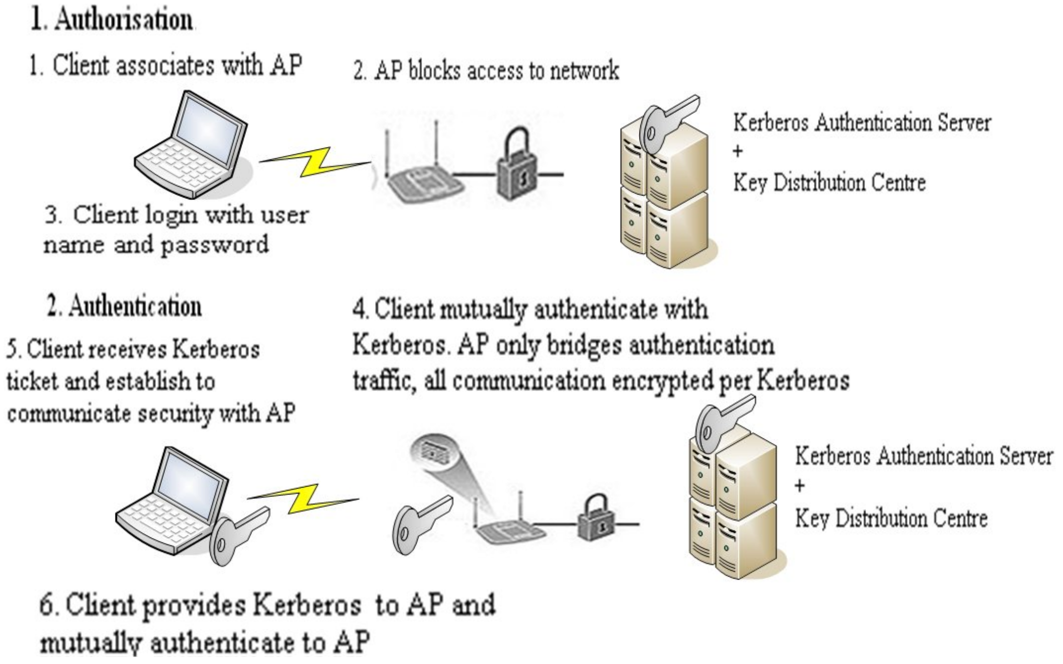

1. Authorisation
1. Client associates with AP
2. AP blocks access to network
Kerberos Authentication Server + Key Distribution Centre
3. Client login with user name and password
2. Authentication
4. Client mutually authenticate with Kerberos. AP only bridges authentication traffic, all communication encrypted per Kerberos
5. Client receives Kerberos ticket and establish to communicate security with AP
Kerberos Authentication Server + Key Distribution Centre
6. Client provides Kerberos to AP and mutually authenticate to AP

Figure 2.2: Kerberos in action in a wireless network

In their study, Harbitter and Menascé have drawn attention to the performance evaluation of Kerberos security protocol in two different achievements, public key assistance and the addition of a proxy server [Harbitter and Menascé, 2002].

Firstly, they used public-key infrastructures *Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)*, *Public Key Cryptography for Cross-Realm Authentication in Kerberos (PKCROSS)* and *Public Key Utilising Tickets for Application Servers (PKTAPP)*. In *PKINIT*, messages are added to change user secret key authentication to public key authentication. It manages secret keys for large number of clients. Nevertheless, it does not address key management of large number of realms. Additionally, as mentioned above, Kerberos uses key distribution and all tickets in its realm are issued by KDC. Since all authentications pass through the KDC, this causes performance bottleneck. At this point, *PKTAPP* is used for trying to eliminate bottleneck and reduce communication traffic by implementing authentication exchange directly between client and application server.

Secondly, in the same study they have proposed the use of proxy servers, *Initial Authentication and Pass Through Authentication using Kerberos V5 and GSS-API (IAKERB)* and *Charon* for mobile communication systems. Former one is used as a proxy server, when a client could not establish a direct connection with KDC. Latter one adapts standard Kerberos authentication to a mobile Personal Digital Assistant (PDA) platform. *Charon* uses Kerberos to establish a trust relationship between a user and a proxy. However, as a result, it is possible to say that, although some additional public-key infrastructures are added to various stages of Kerberos, in terms of server and network capacity, they are fully suitable for simpler networks and could not work with more than one application server. In addition to these, a proxy is used to increase encryption process for both client and server; however, it produces delays during the transactions of authentication messages between client and server. Additionally, since wireless network speed increases, the proxy became insufficient and affects the response time.

Kerberos assisted authentication in mobile ad-hoc networks has been created by utilising traditional features of Kerberos [Pirzada and McDonald, 2004]. Their logic appears to lack evidence that the notorious flaws of traditional Kerberos have been addressed in their solution. These flaws include replay attacks and distributed session keys. However, their solution seems to address issues of password guessing attacks.

Considering these findings a new framework and authentication protocols will be modelled and discussed in the following sections 3.2, 4.2, 5.2 and 5.3.

2.3 Analysis and Verification of Authentication Protocols

In this section, for the verification processes of the authentication protocols and capturing their participants, formal methods are analysed to demonstrate the feasibility of these participants and their potentials. This provides a basis in order to extrapolate the possibilities of what an intruder can achieve with a certain knowledge, and on which level of the protocol, this is achieved.

2.3.1 Formal Verification Methods

In order to provide security for communication networks, usually cryptographic protocols are used and designed. These are necessary for secure key distribution to provide suitable keys for private or authenticated communications and communication principals. In this manner, authenticated cryptographic protocols are required to assure identity of a party to another one that is communicating within a session. Several works that are commonly found in the literature show that when the protocols are analysed critically, they often fail to provide authentication and secure communication [Lowe, 1996].

Currently, formal methods, mathematically-based techniques for the specification, development and verification of software and hardware systems [Monin, 2001], are spreading through into every phase of protocol development. These phases are classified as design, specification and verification. Because of the rapid evolution of authentication protocols and cryptography mechanisms, formal methods for protocol development are improving as well. The major development in formal methods is verification of protocols. According to Meadows, formal verification techniques are classified as follow [Meadows, 1995], [Roscoe, 2005]:

1. General purpose verification tools: This approach involves the modelling and verifying protocols using specification languages and verification tools.

2. Expert systems: These are developed to investigate different states of a protocol. This technique is based on the state machine model of protocols. In general, expert systems are found as inefficient because of thorough going research and their results are often inconclusive.

Table 2.1: Summary of verification methods

Tools	Verification method	Approach
CSP [Hoare, 1985]	General purpose	FDR Checker
Dolev-Yao [Dolev and Yao, 1981]	Expert systems	State machine
BAN-logic [Burrows *et al.*, 1989]	Modal logic	Modal logic
π-calculus [Milner, 1999]	Algebraic approach	Term re-writing
Spi calculus [Abadi and Gordon, 1999]	Algebraic approach	Equivalences between processes

(a)

Tools	Weaknesses	Strengths
CSP [Hoare, 1985]	Can not be used to capture encryption algorithms	Find more attacks within parallel sessions
Dolev-Yao [Dolev and Yao, 1981]	Does not allow participants to remember state information from one to next	Secrecy only
BAN-logic [Burrows *et al.*, 1989]	Hosts excessively trusted	Analysis of knowledge
π-calculus [Milner, 1999]	Indeterminate state machines	Secrecy only
Spi calculus [Abadi and Gordon, 1999]	Minimal intruder capabilities	Proves more than secrecy

(b)

3. Modal logic: Modelling and verifying protocols using modal logics developed for analysis of knowledge and belief during the execution of distributed algorithm.

4. Algebraic approach: This approach is based on modelling of the properties of cryptographic protocols as an algebraic term.

The emphasis on verifying the correctness, or otherwise, the strength of cryptographic protocols has shifted from the error-prone informal approaches to the more acceptable formal approaches. The formal techniques have yielded several tools and procedures and will continue to do so in the immediate foreseeable future. Table 2.2a and 2.2b summarise the developments in verification protocols.
As there is no explicit modelling of an intruder and intruder capabilities in algebraic approach, and principals of a wireless system exposes secrets of the system in modal logic, one

of the most preferred methods is the use of general purpose verification tools. CSP is one of these tools. This is an abstract language designed specifically for the description of communication patterns of concurrent system components that interact through message passing [Hoare, 1985]. The aim of CSP approach is to reduce questions about security protocols and properties to questions concerning whether CSP processes satisfy particular CSP specifications [Schneider, 1998]. This approach forces the separation of properties and protocols, and allows discussion of what is meant by particular kinds of security properties, independent of the protocols that are intended to achieve them. In other words, CSP is particularly suitable for describing protocols close to the level we think of them. The language of CSP is used to present authentication protocol models. Schneider states that formalisation of the protocol into CSP exhibits issues and forces design decisions that may not have been distinctly stated in the original protocol description [Schneider, 1999]. In order to model protocols, the participants in the protocols are modeled as well [Eneh *et al.*, 2006], [Roscoe, 2005], [Schneider, 1998]. In a simple protocol, it is assumed that there are two communicating principals, A and B and an adversary who is the attacker. This is given as follows:

1. With unknown number of clients:
   NET = $(|||_{j} \in_{USER} USER_{j})$ | [trans, rec] | ATTACKER
2. With only two participants (client/agent):
   NET = $(USER_{A}\ |||\ USER_{B})$ | [trans, rec] | ATTACKER

The above descriptions are same, with different number of participants. The second one implies that users A and B inadvertently communicate with the attacker through transmission and reception channels represented as *trans* and *rec* respectively. The medium consists of two channels, one for *transmission* and one for *reception*. Communications are modelled by two types of events, A transmission event is of the form *trans.i.j.m* and is interpreted as "user *i* sends a message *m* destined for user *j*". A reception event is of the form *rec.i.j.m* and means "*i* receives a message *m*, from user *j*" [Dutertre and Schneider, 1997]. The attacker is modelled in a way to have the capacity of intercepting messages in all directions, modifying messages, injecting new messages and transmitting messages [Schneider, 1998]:

$$\text{ATTACKER sat (INIT} \cup (\text{tr} \Downarrow \text{trans})) \vdash \text{tr} \Downarrow \text{rec}$$

The sets of all the messages that pass through the *rec* channel are a function of the initial knowledge of the attacker and the sets of the messages input on the *trans* channel follow:

$$\text{ATTACKER(S)} = \text{trans?i?j?m} \Rightarrow \text{ATTACKER (S} \cup \text{m)} \ \square\ \text{i, j} \ \epsilon\ \text{USER, S}\vdash\text{m rec.i!j!m} \Rightarrow \text{ATTACKER (S)}$$

The above descriptions of `NET` and `ATTACKER` are related with possible deductions and inductions theorems of [Roscoe, 1995], [Paulson, 1998] and [Bella, 2000]. The above descriptions' initial propositions that concern initial knowledge are adapted from [Lowe, 1997b]. The propositions are stated as:

1. $\forall$ m $\in$ INIT.I(m)
2. ($\forall$ m' $\in$ S.I(m'))$\wedge$ S $\vdash$ m $\Rightarrow$ I(m)

These two conditions imply that if the attacker ever knows the messages that satisfy the predicate *I*, then the attacker is able to generate the messages that satisfy *I*. Considering the inference rule [X|Y]π $\Rightarrow$ M, [Cousot and Cousot, 1992], any principal (attacker) that is able to perform deductive or inductive calculation under condition π obtains the set of fact m $\in$M. The rules governing X are according to the theorems for deductions, where the rules for Y is the case for possible inductions.

In the study of [Eneh *et al.*, 2006], a set of messages are defined that depend on the inference rule [X|Y]π $\Rightarrow$ M, [Cousot and Cousot, 1992]. In their study, it is discussed that the capability of induction is such that an attacker can inductively define a member of the set of messages passing through its interface and/or inductively define functions that have been used to create members of message sets. The message sets are defined as:

1. **U** is the universal set comprising of the attackers initial knowledge and sets of the messages passing through the interface where:

   $$\mathrm{U} \subseteq \mathrm{Y}\ \forall\ \mathrm{U} \in \mathrm{X}$$

   The set of deductions X = (INIT $\cup$ (tr$\Downarrow$trans)) $\vdash$ tr $\Downarrow$ rec

2. ω is an operator that builds facts from predicates, the set of rules instances of **P/m** where P $\subseteq$ U and m $\subseteq$ U. m $\in$ M. The rule instance of **P/m** follows the conventional representation of rules such that

   $$\frac{P_1, ..., P_n}{m_1, ..., m_n}(C_1, ..., C_n)$$

   where each P_i (i=1, 2, ..., n) represents the premise(s) C_i (C corresponds to π of the inference rule) the possible side conditions, and m_i, the conclusion. ω implies that if an intruder knows a set of rules for constructing a fact, then the intruder knows the fact. This is expressed in terms of the inference rule of [X|Y]π $\Rightarrow$ M as follows:

$$\omega \Rightarrow \left[\frac{P}{m}\right] \pi \Rightarrow \text{m} \ \forall \ \text{P} \in \text{U and m} \in \text{Y}$$

and

$$\omega(\text{X}) \Rightarrow \left[\frac{P}{m}\right] \pi \Rightarrow \text{m} \ \forall \ \text{P} \in \text{U and m} \in \text{X}$$

3. According to relational set mathematics and the foundation of models in formal verification detailed in [Hoare, 1985] and [Roscoe, 2005], every set has a greatest lower bound $\bot$, then every set has a least upper bound $\top$, and vice versa by symmetry. $\bot \subseteq$ U is the least element in the set U. The significance of $\bot$ lies in the consideration of least fixed points, which is a measure of the *greatest lower bound* and the greatest fixed point (*least upper bound*) of the search space of the set U.

$$\bot \subseteq \text{U} \subseteq \top$$

4. The symbol Θ is a function that constructs a fact from sets, including partially ordered sets or joins. Consider the join:

$$\rho(\rho(\text{U})) \hookrightarrow \rho(\text{U})$$

The induced ordering is as:

$$\text{m} \sqsubseteq \text{y} \overset{def}{=} \text{m} \sqcup \text{y} = \text{y}$$

where, y is a partial order and m is joined to y. In terms of the inference rule $[\text{X}|\text{Y}]\pi \Rightarrow \text{M}$, the function Θ is given by:

$$\text{Y} \triangleq \Theta(\text{X}) = [\rho(\rho(\text{X}))] \ \text{m} \in \text{X} \Rightarrow \text{m}$$

In Section 7.2, an attacker is modelled by using above inductive-deductive theorems.

Apart from these, with the use of the inference rule to analyse a typical Kerberos protocol in the presence of the TGS reveals that the protocol is subject to a TGS masquerade attack. As discussed in the study of [Eneh *et al.*, 2006], authentication in Kerberos requires a client, *C*, to send a request to the authentication server, *AS*, requesting credentials for a given application server, *V*. The AS responds with the requested credentials consisting of a ticket and a session key encrypted with the client's key. Kerberos exchanges may also be in the presence of a TGS. The [Eneh *et al.*, 2006] CSP model of inference is as follows:

1. C → AS : Options $\|$ ID_c $\|$ Realm $\|$ ID_{tgs} $\|$Times $\|$ $Nonce_1$

2. AS → C : Realm $\|$ ID_c $\|$ $Ticket_{tgs}$ $\|$ $E_{kc}[K_{c,tgs}$ $\|$ Times $\|$ $Nonce_1$ $\|$ $Realm_{tgs}$ $\|$ $ID_{tgs}]$

3. C → TGS : Options $\|$ ID_v $\|$ Times $\|$ $Nonce_2$ $\|$ $Ticket_{tgs}$ $\|$ $Authenticator_c^b$

4. TGS → C : $Realm_c$ $\|$ ID_c $\|$ $Ticket_v$ $\|$ $E_{kc,tgs}[K_{c,v}$ $\|$ Times $\|$ $Nonce_2$ $\|$ $Realm_v$ $\|$ $ID_v]$

5. C → V : Options $\|$ $Ticket_v$ $\|$ $Authenticator_c^c$

6. V → C : $E_{kc,v}[TGS_2$ $\|$ Subkey $\|$ $Seq\sharp]$

Nevertheless, the same study shows that, in distributed systems where an intruder has reasonable communication and computational power belonging to the same administrative domain [Eneh *et al.*, 2006], Kerberos may be compromised. In other words, the chance of impersonating a principal by an intruder is higher where AS and TGS are on the same broadcast network.

CSP views principals as processes and was designed for describing systems of interacting components, supported by underlying mathematical theory for reasoning about interacting components [Schneider, 1999]. CSP considers components, in effect processes, as independent self-contained entities with interfaces through which they communicate with their environment. A *trace* is the sequence of communications between the environment and the process [Roscoe, 2005], [Ryan *et al.*, 2000], [Schneider, 1999]. A trace might be finite because the observation was terminated or because the process and environment reach a point where they can not agree on any event. Also, it might be infinite when the observation goes on forever and infinitely many events are transacted. Untimed CSP and Timed CSP are two basic level to record traces. In former one, the events that occur in order are written down. However, the exact times when they happen are written down for the latter one [Roscoe, 2005], [Ryan *et al.*, 2000]. Trace Semantics are used by [Schneider, 1998] to specify security properties for protocols as *trace specifications.* This is done with the following definitions:

$$\text{P } \mathbf{sat} \text{ S} \Leftrightarrow \forall\ \boldsymbol{tr} \in \text{traces(P)} \bullet \text{S}(tr)$$

where *P* is a process and *S* is a predicate. *P* satisfies *S*, if *S(tr)* holds for every trace *tr* of *P*.

In terms of occurrence of events in its traces, the following definition is used for some sets of events R and T:

$$\text{P } \mathbf{sat} \text{ R } \mathbf{precedes} \text{ T} \Leftrightarrow \forall\ \boldsymbol{tr} \in \text{traces(P)} \bullet (tr \restriction \text{R} \neq \langle\rangle \Rightarrow \boldsymbol{tr} \restriction \text{T} \neq \langle\rangle)$$

where a process P satisfies the predicate R precedes T if any occurrences of an event from T is preceded by an occurrence of an event from R in every trace tr of P.

In the same study of [Schneider, 1998] and [Shaikh and Bush, 2006], a set of rules is introduced and defined as well to verify the specifications. According to this study, set of rules defined as **atom** $\boldsymbol{A}$, in this, another three sets are considered which are known as the set of participant identities on the network to be $\boldsymbol{U}$, the set of nonces used by the participants in protocol run as $\boldsymbol{N}$ and the set of encryption keys used as $\boldsymbol{K}$. Altogether, the atoms are defined as $\boldsymbol{A} = \boldsymbol{U} \cup \boldsymbol{N} \cup \boldsymbol{K}$. A message space $\boldsymbol{M}$ contains all the messages and signals that appear during the protocol's run in a way that $\boldsymbol{m} \in \boldsymbol{A} \Rightarrow \boldsymbol{m} \in \boldsymbol{M}$. A rank function ρ is defined in order to map events and messages to integers, ρ: $\boldsymbol{M} \rightarrow \boldsymbol{Z}$. This space is divided into two parts for characterising those messages that an intruder might get hold of [Ryan *et al.*, 2000]:

$$M_P\text{-} = \boldsymbol{m} \in \boldsymbol{M} \mid \rho \leq 0$$
$$M_P\text{+} = \boldsymbol{m} \in \boldsymbol{M} \mid \rho > 0$$

where M_P- is defined as a non positive rank, for those messages that the enemy should never get hold of, where M_P+ is assigned for positive rank for messages that intruder might get hold of without compromising the protocol.

A general *rank function* theorem is presented in order to ensure that a protocol will be verified to be correct with regard to its security properties, if all the steps of the theorem are proven [Schneider, 1998]. The secrecy and authentication properties are concerned with conditions under which particular facts become available to an intruder [Ryan *et al.*, 2000]. For the sets R and T, a rank function, ρ: $\boldsymbol{M} \rightarrow \boldsymbol{Z}$ is such that:

1. $\forall\ m \in \mathbf{IK} \bullet \rho(m) > 0$
2. $\forall\ \mathbf{S} \subseteq M,\ m \in M \bullet ((\forall\ m' \in \mathbf{S} \bullet \rho(m') > 0) \bigwedge \mathbf{S} \vdash m) \Rightarrow \rho(m) > 0$
3. $\forall\ t \in \mathbf{T} \bullet \rho(t) \leq 0$
4. $\forall\ i \in \boldsymbol{U} \bullet \text{User}_i \underset{R}{\|} \text{Stop}$ **maintains** ρ

then NET **sat** R **precedes** T, where NET = $(|||_j \in_{USER} USER_j)$ | [trans, rec] | ATTACKER.

Network, *NET* satisfies R precedes T, shows that each run either satisfies same specifications or else can never perform T. In other words, *NET* which prevents the occurence of R, and then aims to establish that T can not occur [Ryan *et al.*, 2000]. Rank function, ρ is applied not only to messages and signals, but also to events, traces and sets [Schneider, 1999].

In Section 5.4 the importance of rank-functions and construction of these functions according to the CSP rules are presented and discussed.

2.3.2 Model Checking Techniques

The underlying principle of model-checking is the performance of an exhaustive search over behaviours described. FDR, a model-checker, is an automated tool support designed for CSP. FDR is a commercial tool, a product of Formal Systems Ltd. Europe, designed to establish results about concurrent and reactive systems. In addition to permitting safety conditions to be expressible in traces, *failures* and *divergence*, allow for the specification of some set of events a process will eventually be prepared to engage in after a given trace [Ryan *et al.*, 2000]: failures for a condition of nondeterministically arriving at a stable state, and divergence, a condition of unwillingness to arrive at a stable state.

By examination, when used to check security properties, FDR functions by comparing two essential descriptions of systems. These descriptions are the specifications and the implementations. The specification is an abstract and reasonably correct description of the system that is relatively easy to establish, while the implementation is a more complex process with desired efficiencies or structural properties [Ryan *et al.*, 2000], such as parallel composition, concurrency etc. Basically, CSP offers two semantic constructs of interest for FDR to check namely, the denotational and operational constructs. FDR does not manipulate the denotational values directly; rather it exploits the congruencies between the denotational and operational semantics of CSP to calculate the properties based on an operational realisation of the processes in question [Ryan *et al.*, 2000].

The examination done by FDR is such that the implementation is checked to ascertain if the implementation is an exact refinement of the corresponding specification. Refinement is a model checking process that proves the correctness of specifications by using formal methods. Refinement is used to analyse the design in its current stage of the design process [Ryan *et al.*, 2000]. Where the check is successful, it implies that the implementation is a reasonable candidate for substitution into the role of the specification. The prime implication of the stated scenario is that no observation of any run, say of a protocol, can ever suppose that the implementation does not refine the specification; this is accounted for by the exhaustive search performed by model-checking. Nevertheless, where the implementation fails to be a refinement of the specification, it corresponds to a possible attack against the protocol, or system, being described. In cases where FDR finds that implementation does not refine specification, FDR returns a trace of the system that does not satisfy the specification. The returned trace corresponds to an attack upon the protocol. The traces are seen using the debug tool that comes with the FDR distribution, and can be interpreted using interpret that accompanies CASPER, another support tool to be discussed in a later section. Furthermore, the class of systems that can be analysed using model-checking is broadened by using techniques such as data-independence, induction, simplifying transformations etc.

2.3.3 Automated Code Generation

CSP has generated positive interest and astonishing results, especially in verification, by being a reliable assistive technology for the description and analysis of patterns of behaviour of concurrent and real time systems. Despite the unrivalled power of description inherent in the CSP language, the process of producing CSP scripts by hand remained difficult, error-prone, and time-consuming. As a result, a compiler namely, CASPER, has been introduced by [Lowe, 1997], which helps to produce CSP descriptions.

CASPER is a program that automatically produces a CSP description from a more abstract description, thus simplifying the modelling and analysis process. The operation of CASPER is very simple. CASPER takes high level or abstract notation describing protocols to be analysed as input and compiles the notation into the corresponding CSP code as output. The output is then analysed using FDR. In addition, CASPER comes with an interpret utility which is used to interpret the result of the FDR debugger for cases where the implementation failed to be a refinement of the required specification.

A CASPER script could be divided into two parts: a general part that specifies a model of a system running the protocol, and a specific part that defines given functions, the parameters of the protocol. The CASPER input file must define not only the operation of the protocol, but also the system to be checked. Therefore the input file contains two distinct parts:

1. A definition of the way in which the protocol operates, describing the messages passed between the agents, the tests performed by the agents, the types of the data items used, the initial knowledge of the agents and a specification of what the protocol is supposed to achieve.

2. A definition of the actual system to be checked, defining the agents taking part in

the actual system and the roles they play, the actual data-types to be used and the intruder's abilities. [Lowe, 1997]

The first part is a function that returns a model of a system running the protocol and the second part is for defining a particular image of that function by instantiating the parameters of the protocol. The two parts of CASPER are further split into four sections each:

```
script  :: = free-vars-section processes-section
             prot-desc-section spec-section
             act-var-section  [functions-section]
             system-section  intruder-section
```

Each section of the CASPER script has different tasks:
"#Free variables" declares the type of the free variables and functions used in the definition of the protocol.
"#Processes" declares the agents taking part in the protocol and gives information about their state.
"#Protocol description" section defines protocol itself, by giving the messages that run the protocol.
"#Specification" shows the requirements of the protocol.
"#Actual variable" declares the data-types used in the system to be checked with FDR.
"#Functions" gives definitions for the functions used in the protocol.
"#System" defines the system, in terms of the number and types of agents.
"#Intruder" gives the identity and initial knowledge of the intruder.

CASPER is used to demonstrate the feasibility of modelling authentication protocol participants in such a manner to capture their full potentials. This provides a basis in order to extrapolate the possibilities of what an intruder can achieve with a certain knowledge, and on which level of the protocol this is achieved.

2.4 Existing Studies on Performance Evaluation of Security Policies

While the systems are configured to ensure reliable, secure communications, privacy and data integrity, the performance of the underlying networks should also be taken into account. In other words, the systems considered should still be able to perform sufficiently and legitimate access to resources should not be compromised while the security policies are applied

Hence, it is essential to look at possible performance degradation that may be caused by new security measures. This section covers performance evaluation techniques for computer network communications existing in literature as well as performance studies for network security and authentication protocols.

2.4.1 Performance Evaluation Techniques

Performance and availability analysis using modelling, allows engineers, researchers, developers and users to predict and detect possible drawbacks of the systems. This provides early correction and accurate planning. Performance analysis is also useful for optimisation of various system characteristics [Jain, 1991], [Law and Kelton, 2000], [Ever, 2007], [Obaidat and Boudriga, 2010].

There are three techniques used for performance evaluation of computer and communication systems. These are benchmarking, simulation, and analytical modelling [Jain, 1991], [Banks *et al.*, 2005], [Obaidat and Boudriga, 2010].

The process of performance analysis by actual measurements is called *benchmarking*. Benchmarking gives very accurate results. However, benchmarking is only possible if something similar to the proposed system already exists. If the concept considered is new, analytical modelling and simulation are the only techniques available. Benchmarking is also, usually very costly in terms of equipment, personnel, and time [Jain, 1991].

Simulation is the operation of a real world process over time [Banks *et al.*, 2005], [Obaidat and Boudriga, 2010]. Simulation process involves building a simulation model of the system considered. This approach is very flexible, and it gives fairly accurate and acceptable results, but for sufficient accuracy, relatively high computation times are required [Law and Kelton, 2000].

Analytical modelling consists of formulae and/or numerical procedures that are computationally more efficient as compared to simulation [Law and Kelton, 2000], [Banks *et al.*, 2005]. Analytical modelling of a system, or a specific part of it requires less computation. Analytical models generally provide the best information for the effects of various parameters and their interactions [Jain, 1991]. However, analytical modelling requires relatively high level of mathematical skills. Also it sometimes requires a degree of assumptions to simplify the systems considered [Jain, 1991], [Trivedi, 2002], [Banks *et al.*, 2005]. This modelling approach is very widely used in computer science for performance, availability, and reliability evaluation of complex computer and communication systems. This approach is ideal for quick and, once validated for relatively accurate results [Trivedi, 2002], [Banks *et al.*, 2005].

Multi-server computer and communication systems are commonly preferred because of the greater computation power they can provide, and improved reliability. Multi-server systems are more reliable than single server ones, since server failures do not cause complete system failures, and the system can continue serving in a degraded mode. Multi-server system models are extensively used in modelling transaction processing systems, and nodes in communication networks [Chakka and Mitrani, 1994], [Haverkort and Ost, 1997], [Mitrani, 2005]. An analytical modelling approach, *spectral expansion method* is introduced in [Chakka and Mitrani, 1994]. This technique is employed for solution of multi-server communication models. As a result of the popularity of two dimensional processes, various numerical procedures for their steady state analysis have emerged [Haverkort and Ost, 1997], [Mitrani, 2005].

2.4.2 Performance Studies on Network Security

As explained in Section 2.4.1, recent studies show that performance evaluation techniques are employed for various network security protocols as well. As stated in [Zhao and Thomas, 2008], in network communications, authentication is not enough for a system in order to be trustworthy and secure. It is also important being able to respond in a timely manner. This response is not only an obvious usability requirement, but can also be a functional requirement.

Existing studies such as [Harbitter and Menascé, 2002] and [Pirzada and McDonald, 2004] addressed and evaluated the security performance of IEEE 802.11b wireless networks using single server-client architecture and simple traffic models. In [Baghaei and Hunt, 2004] benchmarking experiments, which are based on Windows XP and W2000 Server, are performed in order to investigate the effects of multiple security mechanisms on the performance of multi-client congested and uncongested networks. In the research work of Harbitter and Menascé, the interaction between different security layers and their effects on performance (in terms of response time and throughput) are evaluated. However, their research was limited to a single access point (AP), infrastructure mode and one type of authentication mechanism. For performance evaluation, their results would not be applicable in ad-hoc networks and with use of WPA [Harbitter and Menascé, 2002].

Two mutual authentication and key exchange protocols with anonymity to protect mobile users' privacy in a roaming network environment are proposed and pure performance evaluation measures are presented comparatively with previously presented protocols in [Jiang *et al.*, 2005]. For the first proposed protocol, identity anonymity is achieved by hiding the real user ID based on the secret-splitting principle, while encrypting the real identity with the shared key is used for the second one. The protocols protected a mobile user's privacy in the roaming network environment and reduced the risk that a mobile user uses a compromised session key to communicate with visited networks. The two protocols can be applied depending on the availability of the long-term shared secret key shared by the home network and its users. The performance comparisons have shown significant security improvement [Jiang *et al.*, 2005]. However effects of potential server failures are not taken into account in [Baghaei and Hunt, 2004], [Jiang *et al.*, 2005].

Due to the critical function it provides, the availability of the KDC has a great effect on the performance of the security policy used [Brennen, 2004]. In the study presented in [Zhao and Thomas, 2008] a cost function of a secure KDC is evaluated. A queueing network model has been established. In their study, Zhao and Thomas introduced a new approach to implement a hybrid solution for more efficiency and more accuracy, because of the complex calculation of original equations associated with large scale systems. They have shown how a KDC is modelled by a closed queueing network and found out the service capacity that a KDC satisfies a given number of clients, the optimal number of servers at the KDC and the maximum rate at which keys can be refreshed. Similarly the trade-off between security and performance in considering a model of a KDC is modelled and analysed in [Zhao and Thomas, 2009]. In this study Zhao and Thomas tried to find solution to same problems as stated in [Zhao and Thomas, 2008]. The model is specified using the Markovian Performance Evaluation Process Algebra (PEPA). They implemented a simulation model, attempted to approximate the system behaviour with a much simpler model and also, ordinary differential equation analysis applied since their model suffered from state space explosion and prevented analysis with significant number of clients [Zhao and Thomas, 2009]. In both studies [Zhao and Thomas, 2008] and [Zhao and Thomas, 2009] the availability of the KDC has not been considered. However, assuming that it is unlikely to encounter server failures would be quite optimistic [Trivedi and Xiaomin, 2002].

In the event of KDC failures, administrative functions such as *kadmind* will be unavailable until the primary server is restored or replaced. Specifically, principal management, key creation, and key changes, cannot be done during a primary server failure [Brennen, 2004]. Server failures, affect the overall performance of the KDC, it is desirable to consider a Kerberos server for exact performability modelling.

Considering the above findings analytical models are considered and evaluated in Chapter 6 that check performance and availability.

2.5 Conclusion

In the literature review chapter various existing methods and solution techniques for Kerberos are described and analysed. Basic terms and techniques used in this research are defined. Kerberos authentication protocol and its basic operation in wireless communication networks are studied. The existing authentication methods developed for Kerberos in wireless communication networks are critically analysed and compared. As stated in Sections 2.2.1 and 2.2.2 due to critical nature of wireless communication networks, the existing methods are insufficient to address perspective and requirements in authentication design.

Information on analysis and verification of authentication protocols is given in Section 2.3. Existing methods and techniques are critically analysed. The use of and main relation between these models are underlined. The advantages of using CSP and CASPER/FDR are stated. Also, brief explanations for CSP and CASPER are given to demonstrate the feasibility of modelling authentication protocol participants and to extrapolate the possibilities of what an intruder can achieve with a certain knowledge.

The existing performance evaluation techniques and methods for communication networks are studied and details are given and critically compared. The necessity of having a performance model for network security is stated. Background information of performance models under study for network security is given in detail. As a result, it has been seen that only performance (in terms of response time) measures have been taken into account and availability of components such as KDC have not been considered.

The design of authentication protocols, generally, tends towards the adoption of public key infrastructure methods. This trend is a result of the observed weaknesses and limitations of the shared key schemes. In the context of shared key schemes, compromise of the shared key within any host or principal inadvertently compromises the entire system.

Kerberos is one of the most preferred authentication mechanisms. Kerberos, based on Needham and Schroeder protocols, raises some concerns which include lack of protection against denial-of-service attacks, principals being required to maintain secrecy, and of course, integrity, of secret keys, and clock synchronization to avoid replay attacks. The studies about Kerberos on wireless communication systems, when security provision rely on the 802.1x standard are considered, session hijacking and masquerading attacks are encountered. Recent studies, reveal two categorical enhancements; those that depend on public key infrastructure and those that add a proxy server [Harbitter and Menascé, 2002]. In this study closed networks are used to compare the performance of variants of Kerberos, using a public key infrastructure or adding a proxy server, and concluded that though shared key systems used within proxy servers outperform the public key options they also expose difficulties. Similarly, in [Zhao and Thomas, 2008], it is shown how a KDC is modelled by a closed queueing network and found out the service capacity that a KDC satisfies a given number of clients, the optimal number of servers at the KDC and the maximum rate at which keys can be refreshed. Similarly the trade-off between security and performance in considering a model of a KDC is modelled and analysed in [Zhao and Thomas, 2009]. In this study Zhao and Thomas tried to find a solution to same problems as stated in [Zhao and Thomas, 2008]. In both studies [Zhao and Thomas, 2008] and [Zhao and Thomas, 2009] the availability of the KDC has not been considered. However, assuming that it is unlikely to encounter server failures would be quite optimistic [Trivedi and Xiaomin, 2002].

The emphasis on verifying the correctness, or otherwise, the strength of cryptographic protocols has shifted from the error-prone informal approaches to the more acceptable formal approaches. Formal methods and their related techniques have necessitated emergence of tools and procedures to provide support. CSP, using FDR exposed flaws not discovered by its preceding approaches [Ryan *et al.*, 2000], [Schneider, 1998]. Significantly, the CSP approach shows potentials for adaptation towards the analysis of complex and specialised protocols.

To put it clearly, the aim of this research is to design and model different authentication mechanisms for Kerberos in wireless communication systems. The validation of these models with formal methods and effects on performance degradation are also objectives of the research.

As explicitly discussed, the contributions of the thesis are to provide authentication protocol framework to be used in the design of security mechanisms and to validate the framework by formal verification tools in order to make provision for the formalisation of these approaches and also, combination of both inductive and deductive formal approaches in order to model a stronger attacker model on these modelled authentication protocols.

Depending of this statement the achieved contribution of the thesis is stated as follow in Section 1.3.

Referring to the List of Publications of the Thesis, publication 1 (Kirsal *et. al.* 2005) reveals the work in Chapter Three.
Publication 2 (Kirsal and Gemikonakli(2006)) reveals the work in Chapter Four.
Publication 3 (Kirsal and Gemikonakli(2007a)) reveals the works in Sections 5.2 and 6.2.
Publication 5 (Kirsal and Gemikonakli(2007b)) reveals the work in Section 5.2.1.
Publication 6 (Kirsal and Gemikonakli(2008)) reveals the work in Section 5.3. This work awarded as the best promising research in network security.
Publication 8 (Kirsal and Gemikonakli(2009a)) reveals the work in Section 5.4.
Publication 9 (Kirsal *et. al.* 2009b) reveals the work in Section 6.3.

Chapter 3

A Framework for Solution to the Problem of Trusted Third Party for Wireless Communication Networks

3.1 Introduction

In this chapter, a new framework is proposed in order to provide a background for the design of security solutions for Kerberos security protocol for IEEE 802.11 wireless LANs based on the EAP stack model.

The theoretical grounds of Kerberos, its implications and the capability of the attacker are presented. This chapter is also concerned with the expression of particular security properties and protocols within CSP and FDR, as well as a compiler tool, CASPER that provides a foundation for analysis and verification. Additionally, in terms of authentication and authorisation, security aspects of Kerberos are discussed. Also, this protocol's usability is checked with CASPER.

3.2 The Framework Proposed as a Kerberos Variant

The aim of the proposed framework depicted in Figure 3.1, is to provide a background for security solutions to be employed in wireless LANs where the requirement for security is paramount. The requirement for network security is consistent with permitting authorised access to information and services, while preventing unauthorised access to and corrupting the network.

The framework presented in this section, extends the provisions of [Eneh *et al.*, 2004], critically analyses the existing variants, and provides the avenue by which TAM, presented in Section 2.2.2, [Eneh *et al.*, 2004], is implemented as a variant of Kerberos. This step is necessitated by the need for the TAM framework to efficiently derive the functional benefits of Kerberos. The ultimate goal is to propose a new and improved framework which relies on provisions of the IEEE 802.1x standard. It also uses similar infrastructure components as Kerberos but significantly provides for authentication of the servers.

Following the findings of literature review, since Kerberos is a trusted 3rd party authentication protocol and application independent, its paradigms and entities are finalised. As it can be seen from Figure 3.1, the crytographic protocol, programs and data containing the credentials of the legitimate entities of a particular wireless LAN environment are installed on each of the entities as well as TGS and KDC. The credentials are the identities of the devices (such as MAC addresses) and they are stored with cryptographic protection. The crytographic protocol adopts the challenge-response paradigm [Klensin *et al.*, 1997]. The interactions between the entities are represented using numbers 1-19. These numbers represent the interactions between the legitimate entities of a wireless LAN environment.

Numbers 1, 2, 3, 16, 17 represent the interactions between the client and the access point while numbers 4, 15, 18, 19 represent the interactions between the access point and the authentication server. The numbers 5, 6, 7, 8 and 9, 10, 11, 12 represent the interactions between authentication server and KDC, and authentication server and TGS respectively. Also, numbers 13, 14 represent the interactions between KDC and TGS. The involved steps are explained below:

1. The supplicant (wireless client) sends an Extensible Authentication Protocol over LAN (EAPOL) start message to the authenticator (access point) requesting authentication.

2. The access point (AP) responds with a challenge to the supplicant to supply the supplicant's device identity. The AP also bundles the MAC address of the AP itself along with the challenge on actual network traffic under strong encryption to the supplicant.

3. The supplicant responds to the AP after processing the challenge. The supplicant processes the challenge by decrypting the challenge text and ensuring that the AP's MAC address is found in the supplicant's database of possible APs that the supplicant

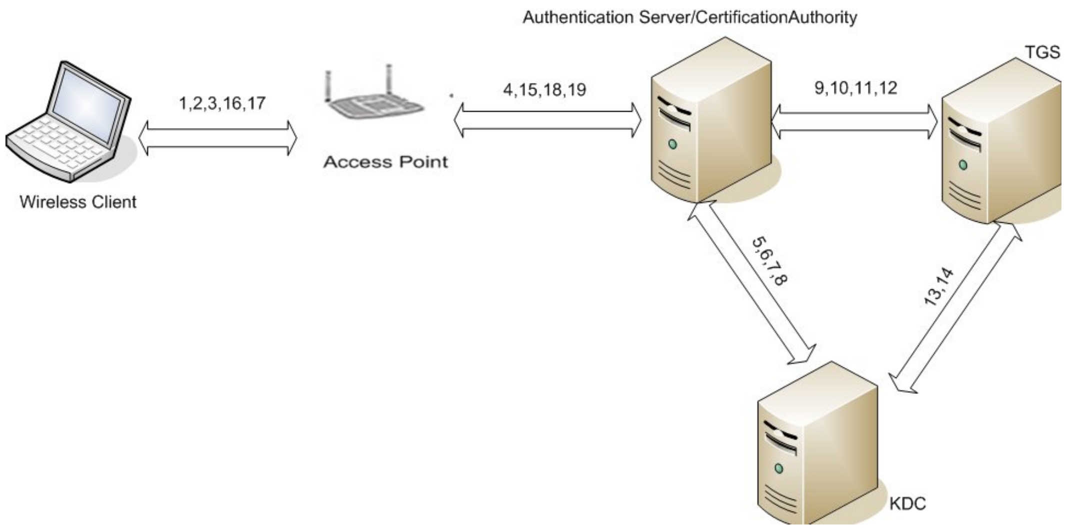


Figure 3.1: The proposed framework

can use to connect to the server or other nodes in the wireless LAN. The supplicant's response is also under strong encryption.

4. The AP challenges the authentication server (AS). The challenge text is bundled with the AP's and the supplicant's MAC address still under strong encryption.

5. The AS sends an EAP Over LANs (EAPOL) start message to the KDC requesting authentication.

6. The KDC responds with a challenge to the AS to supply the AS's device identity. The KDC point also bundles the MAC address of the AS itself along with the challenge on actual network traffic under strong encryption to the AS.

7. The AS challenges the KDC. The challenge text is bundled with the AS's, AP's and the supplicant's MAC addresses still under strong encryption.

8. The KDC responds to the application server's challenge after processing the content of the challenge text. The process involves the decryption of challenge text and a confirmation or proof of knowledge of the existence of both AS and the address of the server. The KDC sends this response under encryption to the AS.

9. The AS sends an EAPOL start message to the TGS requesting authentication.

10. The TGS responds with a challenge to the AS to supply the AS's device identity. The TGS points also bundles the MAC address of the AS itself along with the challenge on actual network traffic under strong encryption to the AS.

11. The AS challenges the TGS. The challenge text is bundled with the AS's, AP's and the supplicant's MAC addresses still under strong encryption.

12. The TGS responds to the AS's challenge after processing the content of the challenge text. The process involves the decryption of challenge text and a confirmation or proof of knowledge of the existence of both AS and the address of the server. The TGS sends this response under encryption to the AS.

13. The KDC server challenges the TGS. The challenge text is bundled with the KDC's, AP's and the supplicant's MAC addresses still under strong encryption.

14. The TGS responds to the KDC's challenge after processing the content of the challenge text. The process involves the decryption of challenge text and a confirmation or proof of knowledge of the existence of both KDC and the address of the KDC. The TGS sends this response.

15. The AS responds to the AP's challenge after processing the content of the challenge text. The processing involves the decryption of the challenge text and a confirmation or proof of knowledge of the existence of both the AP and the supplicant within the secured database of the server. The response includes the MAC address of the server. The AS sends this response under encryption to the AP.

16. The AP challenges the supplicant to run the program to authenticate the end user.

17. If the user responds correctly to the authentication request, the supplicant responds accordingly to the AP.

18. The AP sends the users sign-on response from the supplicant to the AS for the necessary processing.

19. The AS responds to the AP with either the ACCEPT packet or the REJECT packet, depending on the outcome of the processing, to the AP. This makes the AP to transition to the authorised state to allow traffic to and from the supplicant with the ACCEPT message or unauthorised state with the REJECT message.

The proposed model presented above is addition of a new variant on highly confidential, popular authentication protocol, Kerberos for wireless LANs. The full implementation of the model is expected to improve security in terms of authentication and masquerading attacks. Next Section, 3.3 provides the foundation of a design that is developed for analysing, testing and verification of the framework.

3.3 Analysis and Verification of the Designed Framework

This section is concerned with the framework's availability in terms of authentication. Kerberos implications and the capability of the attacker under assumptions of possible deductions with inductive capability in CASPER/FDR are also presented.

The aforementioned two authentication protocols, Kerberos [Kohl and Neuman, 1993], [Neuman and Ts'o, 1994] and Encrypted Key Exchange [Diffie and Hellman, 1976], are combined for the verification. Depending on paradigms and entities (such as being a timed authentication protocol) of the framework and the aforementioned protocols, combined protocol script is written in CASPER. The following script is the part of this protocol:

```
#Processes
INITIATOR(A,na) knows SKey(A), PK,SK(A), passwd(A,B)
RESPONDER(B,S,nb,ns) knows  SKey(B), PK,SK(B), passwd(A,B)
SERVER(S,kab) knows SKey,  PK, SK, passwd

#Protocol description
0. -> A : B
[B != A]
1. -> A : S
2. A -> S : B
3. S -> A : {{ts, B, kab}{SKey(A)}}{passwd(A,B)}
4. S -> A : {{ts, A, kab}{SKey(B)}}{passwd(A,B)} % enc
5. A -> B : enc % {{ts, A, kab}{SKey(B)}}{passwd(A,B)}
6. A -> B : {A, ta, na}{kab}
7. B -> A : {ta, na, nb}{kab}
8. A -> B : {nb, ta}{kab}

#Intruder Information
Intruder = Mallory
IntruderKnowledge = {Alice, Bob, Mallory, Sam, Nm, PK, SK(Mallory), \
SKey(Mallory),passwd(Mallory,Alice), passwd(Mallory,Bob), \
passwd(Alice,Mallory),passwd(Bob,Mallory),passwd(Mallory,Mallory)}
Guessable = Password
```

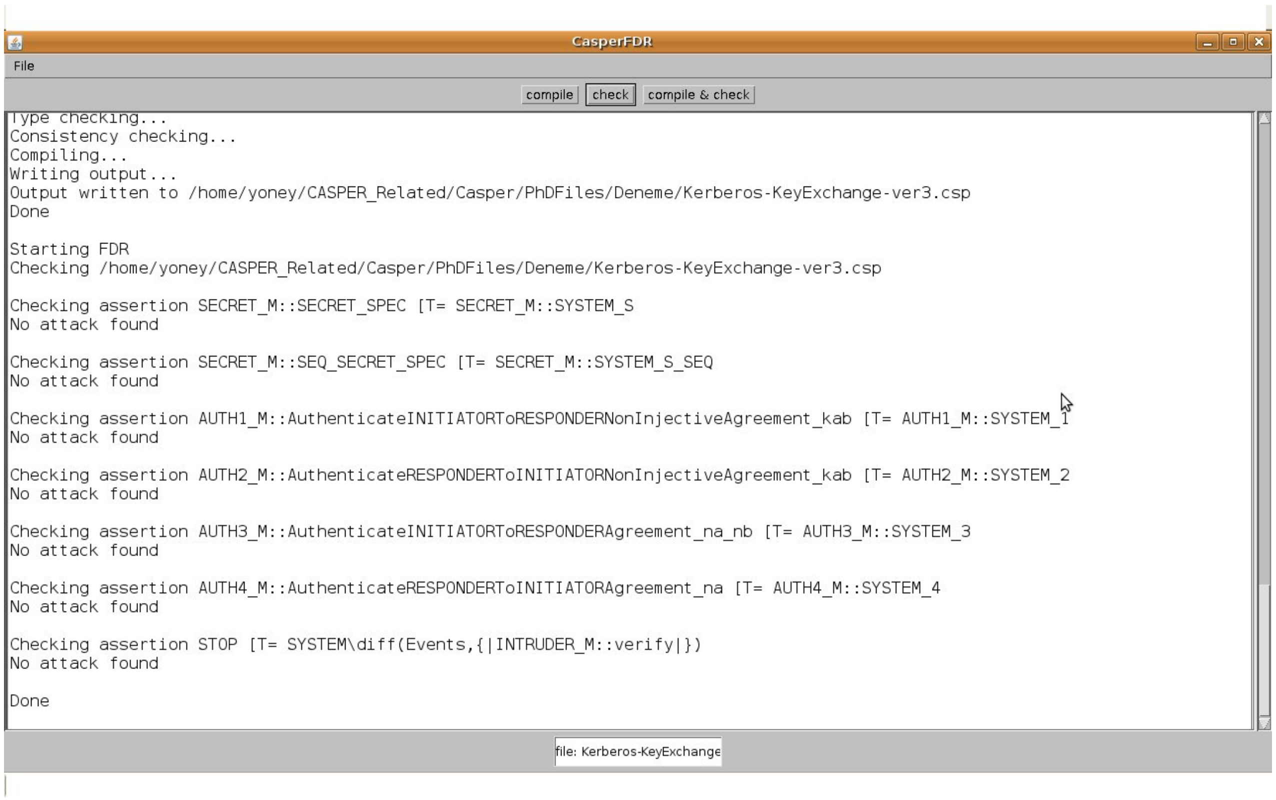
In the "#Processes" section, the first parameter of each *process* (here *A, B* and *S*) should represent agent identities used in the "#Protocol description" section. Names are given to the roles played by the different agents. In this protocol, *INITIATOR, RESPONDER*, and

SERVER are chosen. The parenthesized parameters and the variables following the keyword "*knows*" define the knowledge that the agent is expected to have at the beginning of the protocol run. In the above proposed protocol, the initiator *A* and *S* are expected to know own **identity** *A, S*, the **nonce** *na, nb* and *ns*, the **public key** *PK*, own **secret key** *SK(A)*, **server key** *SKey(A)* and common **password** *passwd(A,B)*. Since, actual work of *S* is designed as SERVER, it is not necessary to define its keys in both INITIATOR and RESPONDER.

In the "#Protocol description" section, the protocol itself is defined by listing the steps in order. These steps will correspond to protocols messages. *Message 0* is used to start the protocol off and tells *A*, the identity of the agent with whom he should run the protocol. Since the above protocol is timed protocol, **timestamps** *ta, ts* are used as variables in the message. Furthermore, the entities of the messages are encrypted with the *server key, SKey*, and the messages are encrypted with their common *passwd(A,B)*, which is distributed by the server. This increases the time it takes for the intruder to decrypt the message. Especially in *Message 4*, the *sender, A* creates and sends message, but the *receiver, B* stores this message in **variable** *enc*, without trying to interpret it. It prefers to wait until it is sure of the authenticity of the communicating agent. In other words, *B* decrypts this message and performs the appropriate checks only after receiving *Message 5*. *Message 4* was encrypted with the inverse of the key receiving in *Message 5*, which *B* expects to be A's password.

The "#Intruder Information" section, is the part of the script that, an intruder's identity and the set of data values that knows, are initially mentioned. The *"Intruder knowledge"* section holds the identifiers and functions of the protocol that are known and can be applied to any other value to those identifiers and functions.

The protocol run through CASPER and checked through FDR. Due to the authentication specifications introduced, there were *no attack found.* The verification of the framework and its authentication specifications are represented in Figure 3.2. The examination of verifications is done by FDR as well in order to check ascertains. The implementation is an exact refinement. The check is successful, where the prime implication of the stated scenario is that no possible attacks found against the protocol.



```
Type checking...
Consistency checking...
Compiling...
Writing output...
Output written to /home/yoney/CASPER_Related/Casper/PhDFiles/Deneme/Kerberos-KeyExchange-ver3.csp
Done

Starting FDR
Checking /home/yoney/CASPER_Related/Casper/PhDFiles/Deneme/Kerberos-KeyExchange-ver3.csp

Checking assertion SECRET_M::SECRET_SPEC [T= SECRET_M::SYSTEM_S
No attack found

Checking assertion SECRET_M::SEQ_SECRET_SPEC [T= SECRET_M::SYSTEM_S_SEQ
No attack found

Checking assertion AUTH1_M::AuthenticateINITIATORToRESPONDERNonInjectiveAgreement_kab [T= AUTH1_M::SYSTEM_1
No attack found

Checking assertion AUTH2_M::AuthenticateRESPONDERToINITIATORNonInjectiveAgreement_kab [T= AUTH2_M::SYSTEM_2
No attack found

Checking assertion AUTH3_M::AuthenticateINITIATORToRESPONDERAgreement_na_nb [T= AUTH3_M::SYSTEM_3
No attack found

Checking assertion AUTH4_M::AuthenticateRESPONDERToINITIATORAgreement_na [T= AUTH4_M::SYSTEM_4
No attack found

Checking assertion STOP [T= SYSTEM\diff(Events,{|INTRUDER_M::verify|})
No attack found

Done
```

Figure 3.2: Verification results of the framework

3.4 Conclusion

In this chapter, a framework is proposed in order to provide a background for design of security solutions for Kerberos security protocol for IEEE 802.11 wireless LANs that require high level of security. The main objective of this proposed model is to identify and develop a new, improved variant for Kerberos.

This chapter is concerned with the expression of particular security properties and protocols within CSP and FDR, as well as a compiler tool, CASPER that provides a foundation for analysis and verification. Additionally, in terms of authentication and authorisation, security aspects of Kerberos are discussed. Also, this protocol's availability is checked with CASPER. The theoretical grounds of a commonly used protocol, Kerberos, its implications and the capability of the attacker under assumptions of possible deductions are presented with inductive capability in CASPER/FDR. Since, possible attacks are minimised, as initial steps for the proposed model, this protocol is a success. New improvements will be introduced on the specifications and description of the protocol in the following chapters.

Chapter 4

Development of an Authentication Protocol to Address the Delayed Decryption Property in Trusted Third Party Authentication Protocols

4.1 Introduction

The work presented in this chapter is a new variant of the proposed authentication protocol presented in Section 3.3. Additionally, the design of this new protocol depends on *delay decryption* property of Kerberos [Neuman and Ts'o, 1994] within the designed framework in Section 3.2. Changes on the protocol and its description are done accordingly.

The design of the protocol, its usability and its verification are the main concerns of this chapter. New entities, as well as their functionalities are defined and introduced. Also, the validation of the use of these entities is analysed.

4.2 Proposed Authentication Protocol with Delay Decryption Property

Despite the proven protocol description and specifications in Section 3.3 that *no attack found*, threats of penetration and other forms of attacks continuously should be considered, since new forms of *intruders* are developed and created. As discussed earlier in Section 3.3, the proposed protocol is *timed authentication protocol*, where decryption can be delayed. By using the opportunity, *masquerading intruder* is prevented to break the decryption.

In this section, the credentials of the devices that are stored with cryptographic protection and the interactions between the entities represented in Section 3.2 are used. Also, same paradigms and entities of the combined aforementioned protocols are kept. However, protocol description has additional entity,*delay decryption* within its script. This entity is one of the paradigms of timed authentication protocol that timestamps allow for a delay of per time unit per message [Kohl and Neuman, 1993], [Lowe *et al.*, 2009]. The following CASPER script is the part of this new variant:

```
#Processes
INITIATOR(A,S,na) knows SK(A), SPK, SKey(A), PK(A), passwd(A,B)
RESPONDER(B,S,nb) knows  SK(B), SPK, SKey(B), passwd(A,B)
SERVER(S,kab) knows PK, SSK(S), SKey, passwd

#Protocol description
0. -> A : B
[A != B]
1. A -> S : {B}{SKey(A)}
2. S -> A : {S, A, ts, {kab}{PK(A)}, PK(B) % pkb}{passwd(A,B)}
3. A -> B : {A, ts, na, {kab}{pkb % PK(B)}}{passwd(A,B)} % v
[A != B]
4. B -> S : {A}{SKey(B)}
5. S -> B : {S, B, PK(A) % pka}{passwd(A,B)}
[decryptable(v, pka) and nth(decrypt(v,pka), 1) == A \
 and nth(decrypt(v,pka), 2) == now \
 and decryptable(nth(decrypt(v,pka), 3), passwd(A,B)) \
 and decryptable(nth(decrypt(v,pka), 4), SK(B))]
<na := nth (decrypt (nth(decrypt(v,pka), 3))) ; \
kab := nth (decrypt (nth(decrypt(v,pka), 4), SK(B)), 1)>
6. B -> A : {nb,na,tb}{kab}
7. A -> B : {nb,ta}{kab}

#Specification
Agreement(A, B, [na])
Secret(A, passwd(A,B), [B])
Secret(A, kab, [B])
Secret(B, kab, [A])

#Intruder Information
Intruder = Mallory
```

```
IntruderKnowledge = {Alice, Bob, Mallory, Sam, Nm, PK, SPK, \
SK(Mallory), SKey(Mallory),passwd(Mallory,Alice), passwd(Mallory,Bob), \
passwd(Alice,Mallory), passwd(Bob,Mallory), passwd(Mallory,Mallory)}
Crackable = SessionKey
Crackable = ServerKey
Crackable = Password
```

In the above protocol, the "# Processes" section shows similarities to Section 3.3's "# Processes" section. However, due to the design and the characteristics of the *protocol description*, elements and entities are varied accordingly to fulfill assigned roles.

In addition to the "# Processes" section, "# Protocol description" and protocol's entities are extended in order to fulfill the *delay decryption* property of the suggested authentication protocol. As it is emphasised in Section 3.3 the protocol's messages are listed in execution order. *Message 0* to *Message 3* are the first part of the protocol which is same as "# Protocol description" of the designed framework in Section 3.3. The second part starts with *Message 5*. In testing *Message 5*, "*decryptable*", "*decrypt*" and "*nth(,n)*" functions that are provided by CASPER are used [Lowe, 1997]. The function "*decryptable*" takes a message and a key, and tests whether the message is encrypted with the inverse of the key. However, the function "*decrypt*" takes a message and a key and decrypts the message with the key. "*nth(,n)*" returns the nth field from the message.

Due to the use of the *delay decryption*, *B* cannot automatically extract any fields from *Message 3*, so more assignments are needed in the *delay decryption* model. The assignments are added by using the functions encapsulated between < >. The first assignment assigns the **nonce**, *na* as the third field of *Message 3*, but the message itself, is encrypted with the common *passwd(A,B)*, which is distributed by the server, has to be decrypted using the inverse of this key which is itself. The second assignment assigns the **session key**, *kab* as the *fourth* component of *Message 3* but the first field of the message is encrypted with B's public key. In addition to these, *delay decryption* is very sensitive to order of fields in the message. That is to say; in the assignment < kab := nth (decrypt (nth(decrypt(v,pka), 4), SK(B)), **1**)>, if number 1 is changed to any other number the output will show a compilation error.

Furthermore, as *nonces* are also being used for authentication between the *agents A* and *B*, the *intruder* can try to make the first attempt to attack *Message 3* when *A* sends **nonce**, *na* to *B*. Due to being a timed authentication protocol, an agent's chance to attempt to connect will be timed-out by the server because of unsuccessful connection attempts thus preventing the attack.

As explained in Section 2.3.3, the "#Specification" section is used to specify the requirements of the protocol, each line represents a particular environment.

- `Secret(A, passwd(A,B), [B])` specifies that any completed run, A can expect the value of the **password**, *passwd(A,B)* to be a secret; B represents the role with whom the secret is shared. Even if the *intruder* cannot masquarade B, if it obtains the value of *passwd(A,B)* assigned by A this specification would fail. `Secret(A,kab, [B])` and `Secret(B,kab, [A])` are similar, where the value that is to be a secret is **SessionKey**, *kab*.

- `Agreement(A, B, [na])` specifies that A is correctly authenticated to B, and the agents agree upon **nonce**, *na*. If B completes a run of the protocol with A, then both agents agreed as which roles they took and the value of the *na*.

Referring to the information from Section 3.3, the "#Intruder Information" section, holds the identifiers and functions of that are known to the protocol and also, other values that can be applied to these identifiers and functions.

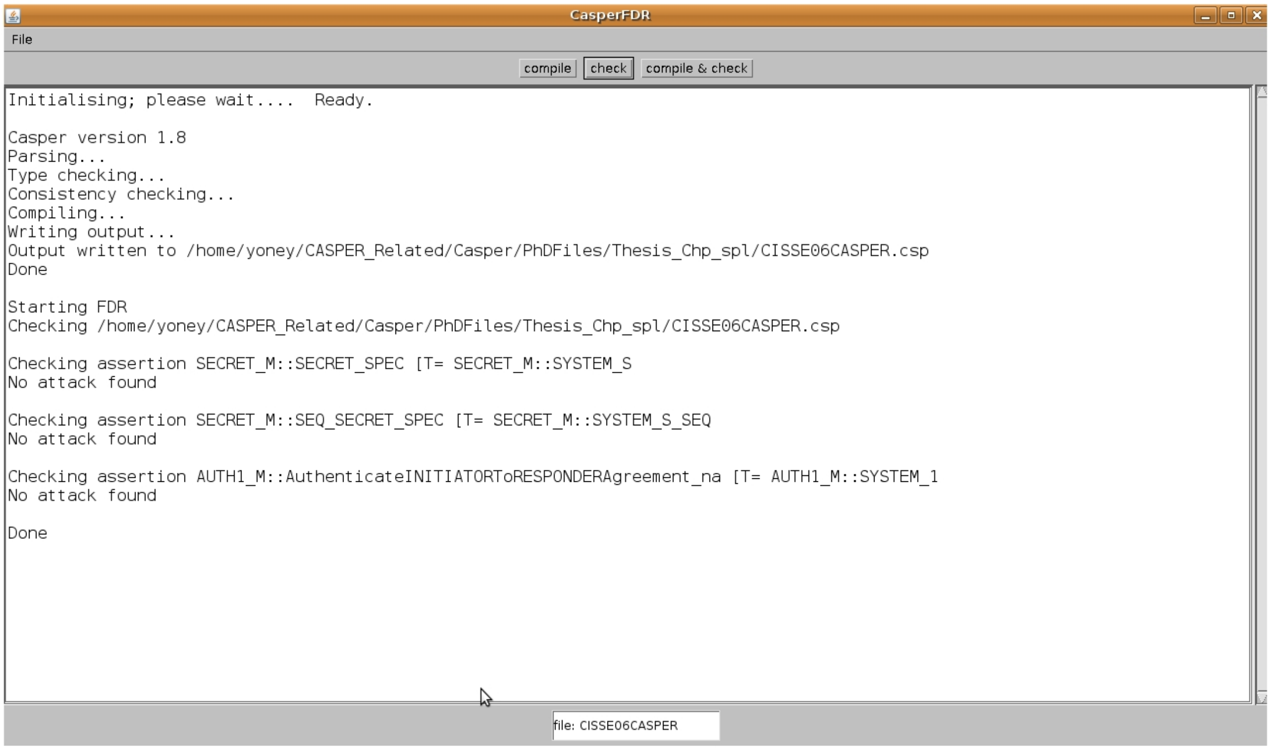
In the following section, the protocol is run through CASPER and checked through FDR. The verification of the protocol, its *delay decryption* property and authentication specifications are also represented through CSP.

4.3 Analysis and Verification of the Designed Authentication Protocol

In this section, the analysis and verification are done through the use of CASPER/FDR and CSP respectively.

4.3.1 Analysis through Code Generation, CASPER

The designed protocol given in Section 4.2 is checked through CASPER for critical analysis to find out errors associated with the design of the protocol for unassailable attacks. Figure 4.1 demonstrates the feasibility of the protocol and the achievements of an *intruder* with knowledge.



```
Initialising; please wait....  Ready.

Casper version 1.8
Parsing...
Type checking...
Consistency checking...
Compiling...
Writing output...
Output written to /home/yoney/CASPER_Related/Casper/PhDFiles/Thesis_Chp_spl/CISSE06CASPER.csp
Done

Starting FDR
Checking /home/yoney/CASPER_Related/Casper/PhDFiles/Thesis_Chp_spl/CISSE06CASPER.csp

Checking assertion SECRET_M::SECRET_SPEC [T= SECRET_M::SYSTEM_S
No attack found

Checking assertion SECRET_M::SEQ_SECRET_SPEC [T= SECRET_M::SYSTEM_S_SEQ
No attack found

Checking assertion AUTH1_M::AuthenticateINITIATORToRESPONDERAgreement_na [T= AUTH1_M::SYSTEM_1
No attack found

Done
```

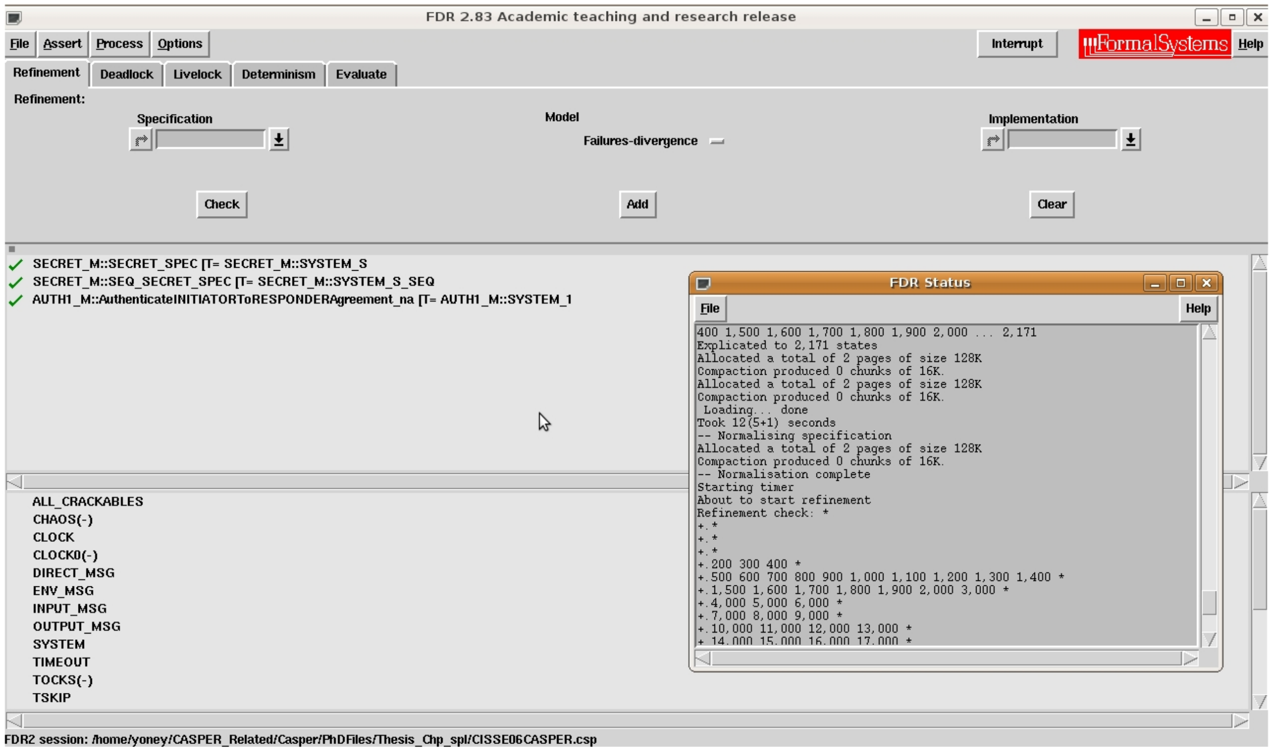
Figure 4.1: Analysis of the new variant of the designed protocol

When the protocol is checked through FDR, due to the *delay decryption* introduced, there were *no attack found*. The ascertains are identified by a tick. The ticks next to the ascertains shown in Figure 4.2 indicated that the design is an exact refinement and successful, since there are no attacks against the protocol and its specifications that are mentioned in Section 4.2.

4.3.2 Verification through Formal Methods, CSP

In this section the CSP representation of the proposed protocol and specified authentication property is modelled. While modelling the different processes of a protocol, advantage of the extensibility of CSP gives the opportunity to add additional elements to the processes. The following scripts are representations of three *participants*, *INITIATOR, RESPONDER, SERVER* of the proposed protocol that are defined in the "#Processes" section of the protocol description in Section 4.2.

```
INITIATOR(A, S, na) =
    [] B : Agent @ A != B & env_I.A.(Env0, B,<>) ->
    output.A.S.(Msg1, Encrypt.(SKey(A), <B>),<>) ->
```



```
✓ SECRET_M::SECRET_SPEC [T= SECRET_M::SYSTEM_S
✓ SECRET_M::SEQ_SECRET_SPEC [T= SECRET_M::SYSTEM_S_SEQ
✓ AUTH1_M::AuthenticateINITIATORToRESPONDERAgreement_na [T= AUTH1_M::SYSTEM_1
```

```
400 1,500 1,600 1,700 1,800 1,900 2,000 ... 2,171
Explicated to 2,171 states
Allocated a total of 2 pages of size 128K
Compaction produced 0 chunks of 16K.
Allocated a total of 2 pages of size 128K
Compaction produced 0 chunks of 16K.
 Loading... done
Took 12(5+1) seconds
-- Normalising specification
Allocated a total of 2 pages of size 128K
Compaction produced 0 chunks of 16K.
-- Normalisation complete
Starting timer
About to start refinement
Refinement check: *
+.*
+.*
+.*
+.200 300 400 *
+.500 600 700 800 900 1,000 1,100 1,200 1,300 1,400 *
+.1,500 1,600 1,700 1,800 1,900 2,000 3,000 *
+.4,000 5,000 6,000 *
+.7,000 8,000 9,000 *
+.10,000 11,000 12,000 13,000 *
+.14,000 15,000 16,000 17,000 *
```

```
ALL_CRACKABLES
CHAOS(-)
CLOCK
CLOCK0(-)
DIRECT_MSG
ENV_MSG
INPUT_MSG
OUTPUT_MSG
SYSTEM
TIMEOUT
TOCKS(-)
TSKIP
```

FDR2 session: /home/yoney/CASPER_Related/Casper/PhDFiles/Thesis_Chp_spl/CISSE06CASPER.csp

Figure 4.2: Analysis of specifications of the new variant of the designed protocol

```
    [] kab : SessionKey @ [] ts : TS @ [] pkb : addGarbage_(PublicKey) @
      input.S.A.(Msg2, Encrypt.(passwd(A, B), <S, A, Timestamp.ts,
      Encrypt.(PK(A), <kab>), pkb>),<>) ->
    output.A.B.(Msg3, Encrypt.(passwd(A, B), <A, Timestamp.ts,
    na, Encrypt.(pkb, <kab>)>),<>) ->[] nb : Nonce @ [] tb : TS @
      input.B.A.(Msg6, Encrypt.(inverse(kab), <nb, na, Timestamp.tb>),<>) ->
    [] ta : TS @
      output.A.B.(Msg7, Encrypt.(kab, <nb, Timestamp.ta>),<na, kab>) ->
    close.A.INITIATOR_role -> STOP

RESPONDER(B, S, nb) =
    [] A : Agent @
    [] v : addGarbage_({Encrypt.(passwd(A, B), <A, Timestamp.ts, na,
    Encrypt.(pkb, <kab>)>) | A <- Agent, B <- Agent, kab <- SessionKey,
    na <- Nonce, ts <- TS, pkb <- addGarbage_(PublicKey)}) @
      A != B & input.A.B.(Msg3, v,<>) ->
    output.B.S.(Msg4, Encrypt.(SKey(B), <A>),<>) ->
    [] pka : addGarbage_(PublicKey) @ [] now : TS @
      decryptable(v, pka) and nth(decrypt(v,pka), 1) == A  and
      nth(decrypt(v,pka), 2) == now  and decryptable(nth(decrypt(v,pka), 3), passwd(A,B))
       and decryptable(nth(decrypt(v,pka), 4), SK(B)) &
      input.S.B.(Msg5, Encrypt.(passwd(A, B), <S, B, pka>),<Timestamp.now>) ->
    RESPONDER'(B, S, nb, A, v, pka, nth(decrypt(nth(decrypt(v,pka),3))))
RESPONDER'(B, S, nb, A, v, pka, na) =
  RESPONDER''(B, S, nb, A, v, pka, na, nth(decrypt(nth(decrypt(v,pka),4),SK(B)),1))
RESPONDER''(B, S, nb, A, v, pka, na, kab) =
    [] tb : TS @
      output.B.A.(Msg6, Encrypt.(kab, <nb, na, Timestamp.tb>),<>) ->
    [] ta : TS @
      input.A.B.(Msg7, Encrypt.(inverse(kab), <nb, Timestamp.ta>),<na, kab>) ->
    close.B.RESPONDER_role -> STOP

SERVER(S, kab) =
    [] A : Agent @ [] B : Agent @
      input.A.S.(Msg1, Encrypt.(SKey(A), <B>),<>) ->
    [] ts : TS @
      output.S.A.(Msg2, Encrypt.(passwd(A, B), <S, A, Timestamp.ts,
      Encrypt.(PK(A), <kab>), PK(B)>),<>) ->
    input.B.S.(Msg4, Encrypt.(SKey(B), <A>),<>) ->
    output.S.B.(Msg5, Encrypt.(passwd(A, B), <S, B, PK(A)>),<>) ->
    close.S.SERVER_role -> STOP
```

The keywords *input* and *output* are used to define *receive* and *send* application respectively, where *rec* and *trans* keywords are the general definition for this purpose in Section 2.3.1, [Schneider, 1998].

The *delay decryption* property and how it is designed within the protocol is explained under the "#Protocol description" in Section 4.2. The *INITIATOR, A* creates and sends a message, *Message 3*, but the *RESPONDER, B* stores this message in variable *v*, without trying to interpret it. That is to say, *RESPONDER, B* decrypts this message and performs the appropriate checks only after receiving message in the future steps, which is defined as *Message 5*. This is verified with the following CSP script:

```
 A != B & input.A.B.(Msg3, v,<>) ->
    output.B.S.(Msg4, Encrypt.(SKey(B), <A>),<>) ->
    [] pka : addGarbage_(PublicKey) @ [] now : TS
```

Referring the test of *Message 5* "*decryptable*", "*decrypt*" and "*nth(,n)*" functions provided by CASPER are used in Section 4.2. However, more assignments are needed in the *RESPONDER, B*. These can be given as follows:

```
input.S.B.(Msg5, Encrypt.(passwd(A, B), <S, B, pka>),<Timestamp.now>) ->
RESPONDER'(B, S, nb, A, v, pka, nth(decrypt(nth(decrypt(v,pka),3))))
RESPONDER'(B, S, nb, A, v, pka, na) =
RESPONDER''(B, S, nb, A, v, pka, na, nth(decrypt(nth(decrypt(v,pka),4),SK(B)),1))
RESPONDER''(B, S, nb, A, v, pka, na, kab) =
    [] tb : TS @
```

The first assignment, `RESPONDER'(B, S, nb, A, v, pka, nth(decrypt(nth(decrypt(v,pka),3))))` assigns the **nonce** *na* as the **third** field of *Message 3*, but the message itself, is encrypted with the **password**, *passwd(A,B)*, has to be decrypted using the inverse of this key which is itself. The second assignment, `RESPONDER'(B,S,nb,A,v,pka,na) = RESPONDER''(B, S, nb, A, v, pka,na, nth(decrypt(nth(decrypt(v,pka),4),SK(B)),1))` assigns the **SessionKey**, *kab* as the **fourth** component of *Message 3* but the first field of the message is encrypted with B's **public key**, *PK(B)* and decryption has to be done by using the inverse of this key which is **secret key**, *SK(B)* of B.

Since the designed, proposed protocol is time sensitive (i.e timed authentication protocol), introduction of any delay will prevent intruder's attempt to launch an attack. Because of this, the *delay decryption* technique that is used in here is to delay intruders. In the proposed protocol, *RESPONDER, B* should complete a protocol run with *A*. The following CSP process implies that *A* was running the protocol with *B* and there is mutual agreement between them. This mutual agreement depends on the value of **nonce**, *na* chosen:

```
AuthenticateINITIATORToRESPONDERAgreement_na_0(A) =
    signal.Running1.INITIATOR_role.A?B?na ->
    signal.Commit1.RESPONDER_role.B.A.na -> STOP
  AlphaAuthenticateINITIATORToRESPONDERAgreement_na_0(A) =
    {|signal.Running1.INITIATOR_role.A.B,
      signal.Commit1.RESPONDER_role.B.A |
        B <- inter(Agent, HONEST)|}
```

In addition to the *nonces*, *timestamps* are also being used for time agreement between the processes A and B. The system is verified with the following CSP code:

```
SYSTEM
      [[send.A.B.ALGEBRA_M::rmb((Msg7, Encrypt.(kab, <nb, Timestamp.ta>), <na, kab>))
          <- signal.Claim_Secret.A.ALGEBRA_M::applyRenaming(passwd(A, B)).{B},
        send.A.B.ALGEBRA_M::rmb((Msg7, Encrypt.(kab, <nb, Timestamp.ta>), <na, kab>))
          <- signal.Claim_Secret.A.ALGEBRA_M::applyRenaming(kab).{B},
        receive.A.B.ALGEBRA_M::rmb((Msg7, Encrypt.(kab, <nb, Timestamp.ta>), <na, kab>))
          <- signal.Claim_Secret.B.ALGEBRA_M::applyRenaming(kab).{A} |
            A <- Agent_renamed_, B <- Agent_renamed_, na <- Nonce_renamed_,
            kab <- SessionKey_renamed_, nb <- Nonce_renamed_,
            ta <- TS_renamed_
      ]] \ diff(Events,Alpha_SECRETS)
```

The following CSP code represents the "#Intruder Information" section, where *Intruder-Knowledge* definition and its relevant initial knowledge are defined. Detailed explanations are done in Sections 3.3 and 4.2:

```
IK0_init = union({Alice, Bob, Mallory, Sam, Nm, SK(Mallory), SKey(Mallory),
             passwd(Mallory, Alice), passwd(Mallory, Bob),
             passwd(Alice, Mallory), passwd(Bob, Mallory),
             passwd(Mallory, Mallory), Garbage}, TimeStamp)
 INTRUDER_1 =
    (chase(INTRUDER_0)
      [[ hear.m_ <- send.A_.B_.(l_,m_,se_) |
           (l_,m_,se_,re_) <- DIRECT_MSG,
           A_ <- diff(SenderType(l_),{Mallory}), B_ <- ReceiverType(l_) ]]
     [|{| hear |}|] STOP)
      [[ say.m_ <- receive.A_.B_.(l_,m_,re_) |
           (l_,m_,se_,re_) <- DIRECT_MSG,
           A_ <- SenderType(l_), B_ <- ReceiverType(l_) ]]
```

The keywords *hear, say* are used to represent hearing and saying a message during the authentication and transfer of a message across network. `(l_,m_,se_,re_)` is types of *messages* that are *sent* and *received* by processes (agents) as they are considered by those agents.
Addition to the *intruder* definition, the initially known facts are added to the following CSP script:

```
SAY_KNOWN_0 =
    (inter(IK1, ALL_SECRETS_DI) != {} & dummy_leak -> SAY_KNOWN_0)
    [] dummy_send -> SAY_KNOWN_0
    [] dummy_receive -> SAY_KNOWN_0
  SAY_KNOWN =
    SAY_KNOWN_0
      [[ dummy_leak <- leak.f_ | f_ <- inter(IK1, ALL_SECRETS_DI) ]]
      [[ dummy_send <- send.A_.B_.(l_,m_,se_) |
           (l_,m_,se_,re_) <- DIRECT_MSG, components_(m_) <= IK1,
           A_ <- diff(SenderType(l_),{Mallory}), B_ <- ReceiverType(l_) ]]
      [[ dummy_receive <- receive.A_.B_.(l_,m_,re_) |
           (l_,m_,se_,re_) <- DIRECT_MSG, components_(m_) <= IK1,
           A_ <- SenderType(l_), B_ <- ReceiverType(l_) ]]
  STOP_SET = {| send.Mallory |}
INTRUDER =
    (INTRUDER_1 [| STOP_SET |] STOP) ||| SAY_KNOWN

IntruderInterface = Union({{| send, receive |}, {|crack|}})
AlphaSystem = {|env, send, receive, close, tock|}
SystemManagerInterface = inter(AlphaSystem,CRACKING_M::AlphaManager)
SYSTEM = (SYSTEM_M::TSYSTEM_1 [|SystemManagerInterface|] CRACKING_M::Manager)
            [|IntruderInterface|] INTRUDER_M::INTRUDER
```

The keyword *leak* is used to signal that a possible secret has been learnt. Elements such as `f_` , `IK1` are components of the *intruder* for currently unkown *fact* and *initial knowledge.*

4.4 Conclusion

A new variant for the designed authentication protocol represented in Section 3.3 is presented in this chapter. Kerberos and Key-Exchange authentication protocols are combined and the design of this new protocol depends on *delay decryption* property of Kerberos. The main objectives are to propose a protocol, to check its usability and to verify security concerns in terms of authentication. New entities, as well as their functionalities are defined and introduced. Changes on the protocol and its description are done accordingly and analysed within CASPER/FDR. Also, the validation of the use of these entities is analysed within CSP.

Despite the presence of a highly specified intruder and its attack attempts, it is proven that, authentication specifications, design of the secrets in the protocol, parameterised specifications for all secrets, all keys and hash functions in the system and timing functions of the system for secrecy checking are successful in terms of attack prevention.

Chapter 5

Development of a New Solution for Frequent Key Renewal under Pseudo-secure Conditions

5.1 Introduction

This chapter presents an approach of dynamically renewing keys under pseudo-secure situations, thus significantly reducing the chances of potential intruders. The proposed approach involves secure key distributions at various intervals. During key distribution, temporary interruption to link/server access is ensured. The access restriction happens for short intervals. As discussed earlier in Section 1.4, pseudo-secure situations are generated from secure random situations, and that are very hard for an observer to distinguish from true random secure conditions.

An approach that involves a new authentication protocol that combines frequent key renewal with *timed* authentication is also designed. The analysis and verification of authentication properties and results of the designed protocol are presented and discussed.

5.2 New Solution for Frequent Key Renewal with Shut Down Access

Owning to the growing popularity, the design and proposal of authentication approaches for improving security of networks, threats of penetration and various forms of attacks have

continued to evolve. As mentioned earlier in Sections 3.2 and 4.2, preventing intruder access to a network, increasing the decryption time of messages for an attacker is another way of attack prevention. Especially, security protocols on distributed systems are time-sensitive. In the analysis of delayed decryption systems, *timestamps* play an important role.
In this section a new approach is proposed. The proposed model is based on external link(s) which are shut down for short intervals to ensure secure distribution of randomly generated keys, and then the links are re-instated. This will considerably restrict time for intruders to break encryptions. This aims at minimising risks from external sources assuming that it is easier to control legitimate users behaving badly. The proposed system is shown in Figure 5.1. Renewing keys at various intervals while potential intruders are blocked out would inevitably work against intruders.

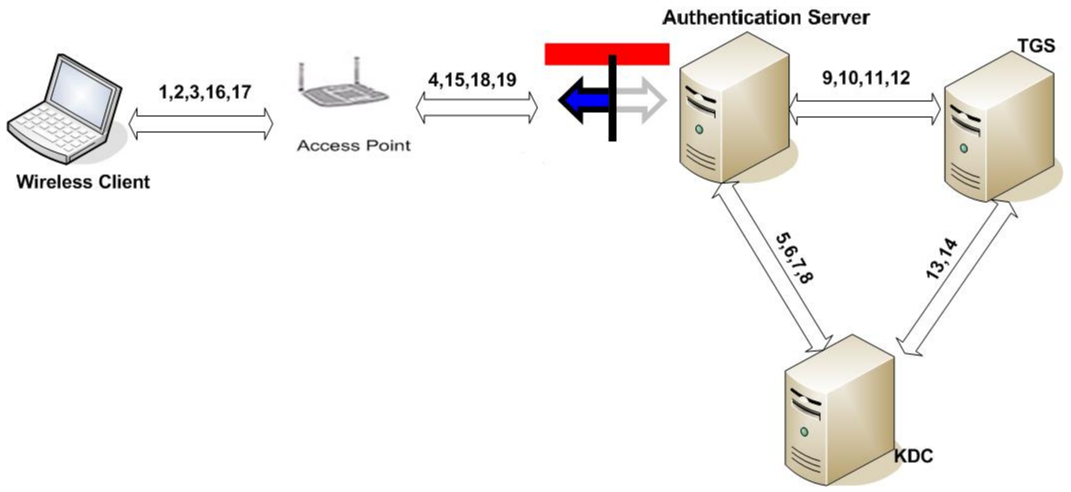


Figure 5.1: The proposed framework with shut-downs

The designed framework adopts the challenge-response paradigm that is defined in Section 3.2. The interactions between the entities and same paradigms of Section 3.2 are adopted as well in this framework. Also, aforementioned protocol description with *delay decryption* of Section 4.2 is improved for renewing keys at various intervals while system is temporarily unavailable to external access for a period of time. The following CASPER script is the part of the designed protocol:

```
#Processes
INITIATOR(A,S,na) knows SK(A), SKey(A), PK, passwd(A,B)
RESPONDER(B,S,nb) knows  SK(B), SKey(B), passwd(A,B)
SERVER(S,kab) knows PK, SKey, passwd

#Protocol description
0. -> A : B
```

```
[A != B]
1. A -> S : {B}{SKey(A)}
2. S -> A : {S, A, ts, {kab}{PK(A)}, PK(B) % pkb}{passwd(A,B)}
3. A -> B : {A, ta, na, {kab}{pkb % PK(B)}}{passwd(A,B)} % v
[A != B]
4. B -> S : {A, tb}{SKey(B)}
5. S -> B : {S, B, ts, PK(A) % pka}{passwd(A,B)}
[decryptable(v, pka) and nth(decrypt(v,pka), 1) == A \
 and nth(decrypt(v,pka), 2) == now \
 and decryptable(nth(decrypt(v,pka), 3), passwd(A,B)) \
 and decryptable(nth(decrypt(v,pka), 4), SK(B))]
<na := nth (decrypt (nth(decrypt(v,pka), 3))) ; \
kab := nth (decrypt (nth(decrypt(v,pka), 4), SK(B)), 1)>
6. B -> A : {nb,na,tb}{kab}
7. A -> B : {nb,ta}{kab}

#Specification
Secret(A, passwd(A,B), [B])
TimedAgreement(A,B,2,[kab])

#Intruder Information
Intruder = Mallory
IntruderKnowledge = {Alice, Bob, Mallory, Sam, Nm, PK, SK(Mallory), SKey(Mallory)}
Guessable = SessionKey
Guessable = Password
```

In the above protocol, the "# Processes" section is same as the Section 4.2's "# Processes" section.

"# Protocol description" and protocol's entities are extended in order to fulfill the *shut down for short intervals to ensure secure distribution of randomly generated keys*, of the suggested authentication protocol. *Message 0* to *Message 3* are considered as the first part of the protocol which is same as "# Protocol description" of the designed protocol in Section 4.2. The second part starts with *Message 5*. In the test of *Message 5* "*decryptable*", "*decrypt*" and "*nth*(,*n*)" functions that are provided by CASPER are used [Lowe, 1997], [Lowe *et al.*, 2009]. The function "*decryptable*" takes a message and a key, tests whether the message is encrypted with the inverse of the key. However, the function "*decrypt*" takes a message and a key and decrypts the message with the key. "*nth*(,*n*)" returns the nth field from the message. Due to *delay decryption* and *shut down* used, *B* cannot automatically extract any fields from *Message 3*, so more assignments are needed in the *delay decryption* model. The assignments are added by using the functions encapsulated between $< >$. The first assignment assigns the **nonce**, *na* as the third field of *Message 3*, but the message itself, is encrypted with the common *passwd(A,B)*, which is distributed by the server, has to be decrypted using the inverse of this key which is itself. The second assignment assigns the **session key**, *kab* as the *fourth* component of *Message 3* but the first field of the message is encrypted with B's public key. In addition to these, *delay decryption* is very sensitive to order of fields in the message. That is to say; if number 1 (written in bold) in the assignment < kab := nth (decrypt (nth(decrypt(v,pka), 4), SK(B)), **1**)> is changed to any other number the output will show a compilation error.

Furthermore, as *nonces* and *timestamps* are also being used for authentication between the *agents A, B* and *S*, the *intruder* can try to make the first attempt to attack in *Message 3* when *A* sends **nonce**, *na* and **timestamp**, *ta* to *B*.

Due to being a timed authentication protocol, an agent's chance to attempt to connect will be timed-out by the server because of unsuccessful connection attempts thus preventing attacks. Also, during the shut downs for short intervals for distribution of randomly generated keys, agents are mutually agreed on unit of a time for the protocol run. These are mentioned in the "#Specification" section that is used to specify these requirements.

- `Secret(A, passwd(A,B), [B])` specifies that any completed run,*A* can expect the value of the **password**, *passwd(A,B)* to be a secret; *B* represents the role with whom the secret is shared. Even if the *intruder* cannot masquerade *B*, if it obtains the value of *passwd(A,B)* assigned by *A* this specification would fail.

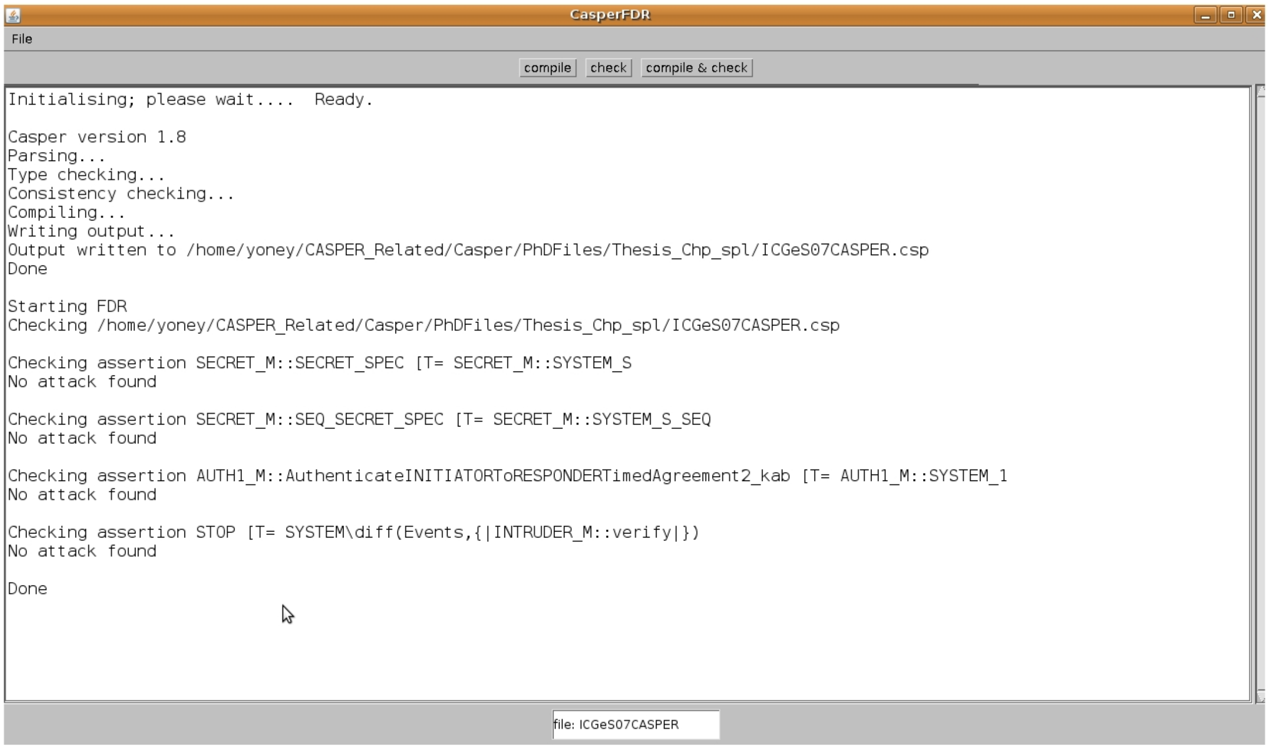
- `TimedAgreement(A, B, 2, [kab])` is a timed version of `Agreement(A,B,[])` that is used in designed protocol in above Section 4.2, specifies that *A* is correctly authenticated to *B*, and the agents agree upon **SessionKey**, *kab*, where *A*'s run is within the *2* **time units** of *B*.

The "#Intruder Information" section, is the part of the script that, an intruder's identity and the set of data values that knows, are initially mentioned. The *"Intruder knowledge"* section holds the identifiers and functions of the protocol that are known and can be applied to any other value to those identifiers and functions. The capability of the attacker under assumptions of possible deductions are presented with inductive capability.

After describing the protocol to achieve improved security for authentication, in the following section, the protocol is run through CASPER and checked through FDR.

5.2.1 Analysis through Code Generation and Model Checking, CASPER/FDR

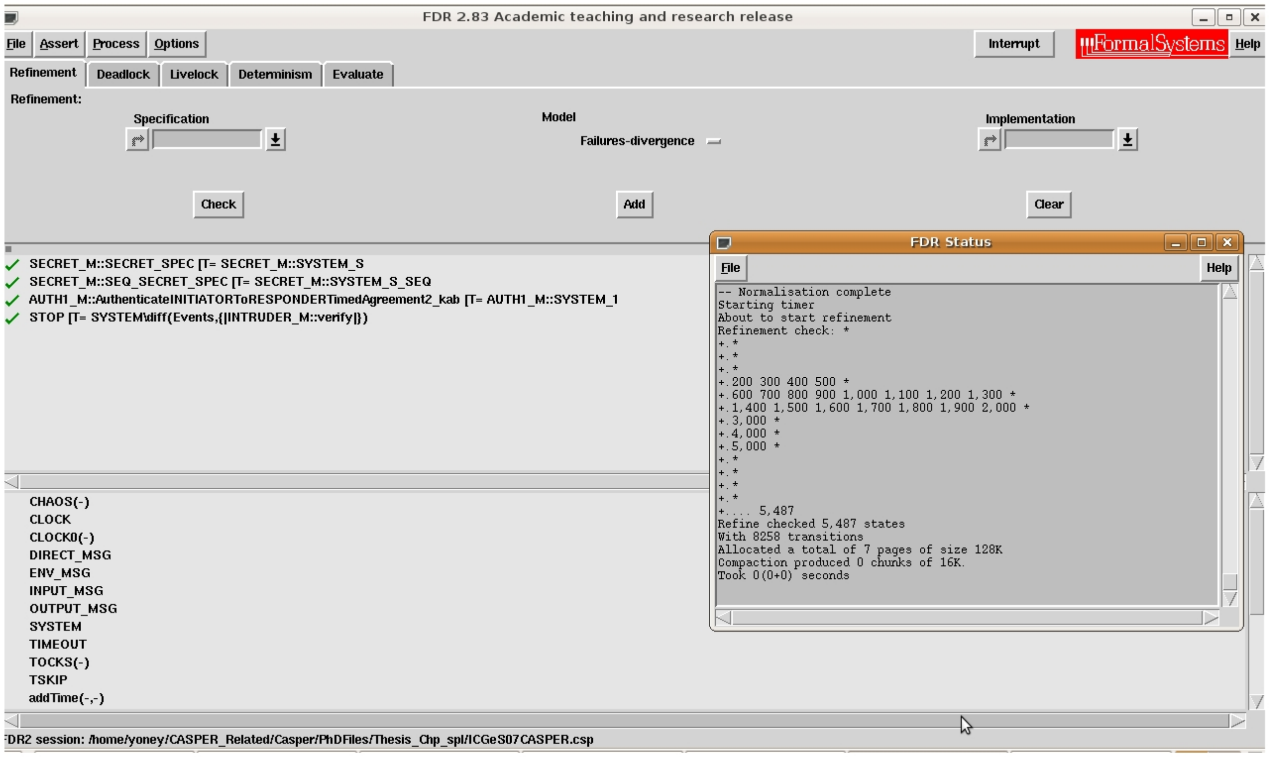
To find out unassailable attacks of the designed protocol in above Section 5.2, critical analysis of the protocol and authentication specifications are done through CASPER/FDR. The following Figure 5.2 demonstrates the feasibility of the protocol and the achievements of an *intruder* with certain knowledge.



```
Initialising; please wait....  Ready.

Casper version 1.8
Parsing...
Type checking...
Consistency checking...
Compiling...
Writing output...
Output written to /home/yoney/CASPER_Related/Casper/PhDFiles/Thesis_Chp_spl/ICGeS07CASPER.csp
Done

Starting FDR
Checking /home/yoney/CASPER_Related/Casper/PhDFiles/Thesis_Chp_spl/ICGeS07CASPER.csp

Checking assertion SECRET_M::SECRET_SPEC [T= SECRET_M::SYSTEM_S
No attack found

Checking assertion SECRET_M::SEQ_SECRET_SPEC [T= SECRET_M::SYSTEM_S_SEQ
No attack found

Checking assertion AUTH1_M::AuthenticateINITIATORToRESPONDERTimedAgreement2_kab [T= AUTH1_M::SYSTEM_1
No attack found

Checking assertion STOP [T= SYSTEM\diff(Events,{|INTRUDER_M::verify|})
No attack found

Done
```

Figure 5.2: Analysis of frequent key renewal with shut-downs

When the protocol is checked through FDR, due to the *shut down access* and *frequent key renewal* introduced addition to the *delay decryption* property and `TimedAgreement(A,B,2,[kab])` specification, there were *no attack found.* The ascertains are checked in Figure 5.3. The design is an exact refinement.The check is successful, where no possible attacks against the protocol.


FDR 2.83 Academic teaching and research release
File Assert Process Options
Interrupt
FormalSystems
Help
Refinement Deadlock Livelock Determinism Evaluate
Refinement:
Specification
Model
Failures-divergence
Implementation
Check
Add
Clear
FDR Status
File
Help

```
SECRET_M::SECRET_SPEC [T= SECRET_M::SYSTEM_S
SECRET_M::SEQ_SECRET_SPEC [T= SECRET_M::SYSTEM_S_SEQ
AUTH1_M::AuthenticateINITIATORToRESPONDERTimedAgreement2_kab [T= AUTH1_M::SYSTEM_1
STOP [T= SYSTEM\diff(Events,{|INTRUDER_M::verify|})
```

```
CHAOS(-)
CLOCK
CLOCK0(-)
DIRECT_MSG
ENV_MSG
INPUT_MSG
OUTPUT_MSG
SYSTEM
TIMEOUT
TOCKS(-)
TSKIP
addTime(-,-)
```

```
-- Normalisation complete
Starting timer
About to start refinement
Refinement check: *
+.*
+.*
+.*
+.200 300 400 500 *
+.600 700 800 900 1,000 1,100 1,200 1,300 *
+.1,400 1,500 1,600 1,700 1,800 1,900 2,000 *
+.3,000 *
+.4,000 *
+.5,000 *
+.*
+.*
+.*
+.*
+.... 5,487
Refine checked 5,487 states
With 8258 transitions
Allocated a total of 7 pages of size 128K
Compaction produced 0 chunks of 16K.
Took 0(0+0) seconds
```

```
FDR2 session: /home/yoney/CASPER_Related/Casper/PhDFiles/Thesis_Chp_spl/ICGeS07CASPER.csp
```

Figure 5.3: Analysis of specifications of frequent key renewal with shut-downs

5.3 Security Aspects of Combined Use of Timed Authentication Protocol and Frequent Key Renewal

This section presents an approach that involves the authentication protocol in Section 5.2 and combines passwords and session keys for authentication purposes. In terms of authentication, this section is concerned with increasing the time for an intruder to break an encryption and hence improve the security. Times of breaking the encryption for the designed protocols defined in Sections 3.3, 4.2 and 5.2 are compared with the new designed protocol presented in this section.

As compared to the protocol in Section 5.2, changes are done in the "# Specification" and "# Intruder Information" sections. The intruder's possible powers of deduction with inductive capability are presented in the following CASPER script:

```
#Processes
INITIATOR(A,S,na) knows SK(A), SKey(A), PK, passwd(A,B)
RESPONDER(B,S,nb) knows  SK(B), SKey(B), passwd(A,B)
SERVER(S,kab) knows PK, SKey, passwd
```

```
#Protocol description
0. -> A : B
[A != B]
1. A -> S : {B}{SKey(A)}
2. S -> A : {S, A, ts, {kab}{PK(A)}, PK(B) % pkb}{passwd(A,B)}
3. A -> B : {A, ta, na, {kab}{pkb % PK(B)}}{passwd(A,B)} % v
[A != B]
4. B -> S : {A, tb}{SKey(B)}
5. S -> B : {S, B, ts, PK(A) % pka}{passwd(A,B)}
[decryptable(v, pka) and nth(decrypt(v,pka), 1) == A \
 and nth(decrypt(v,pka), 2) == now \
 and decryptable(nth(decrypt(v,pka), 3), passwd(A,B)) \
 and decryptable(nth(decrypt(v,pka), 4), SK(B))]
<na := nth (decrypt (nth(decrypt(v,pka), 3))) ; \
kab := nth (decrypt (nth(decrypt(v,pka), 4), SK(B)), 1)>
6. B -> A : {nb,na,tb}{kab}
7. A -> B : {nb,ta}{kab}

#Specification
StrongSecret(A, passwd(A,B), [B])
TimedAliveness(A,B,100)
TimedAgreement(A,B,2,[kab])

#Intruder Information
Intruder = Mallory
IntruderKnowledge = {Alice, Bob, Mallory, Sam, Nm, PK, SK(Mallory), SKey(Mallory)}
Guessable = SessionKey
Crackable = SessionKey
Crackable = ServerKey
Crackable = Password
```

Owing to the fact that the authentication protocol in Section 5.2 is used in this section. The "#Protocol description" and its "#Processes" sections are not modified with new entities.

Due to being a timed authentication protocol and during the shut downs for short intervals for distribution of randomly generated keys, the protocol's "#Specification" section is changed to increase the time for the authentication purposes. These are mentioned in the "#Specification" section that is used to specify these requirements.

- `StrongSecret(A, passwd(A,B), [B])` is similar to `Secret(A, passwd(A,B), [B])` that is used in above Section 5.2, except it also includes incomplete runs. It specifies that any completed run, *A* can expect the value of the **password**, *passwd(A,B)* to be a secret; *B* represents the role with whom the secret is shared. Even if the *intruder* cannot masquerade as *B*, if it obtains the value of *passwd(A,B)* assigned by *A* this specification would fail. This specification is used because the secret, *passwd(A,B)* is significant outside of the protocol.

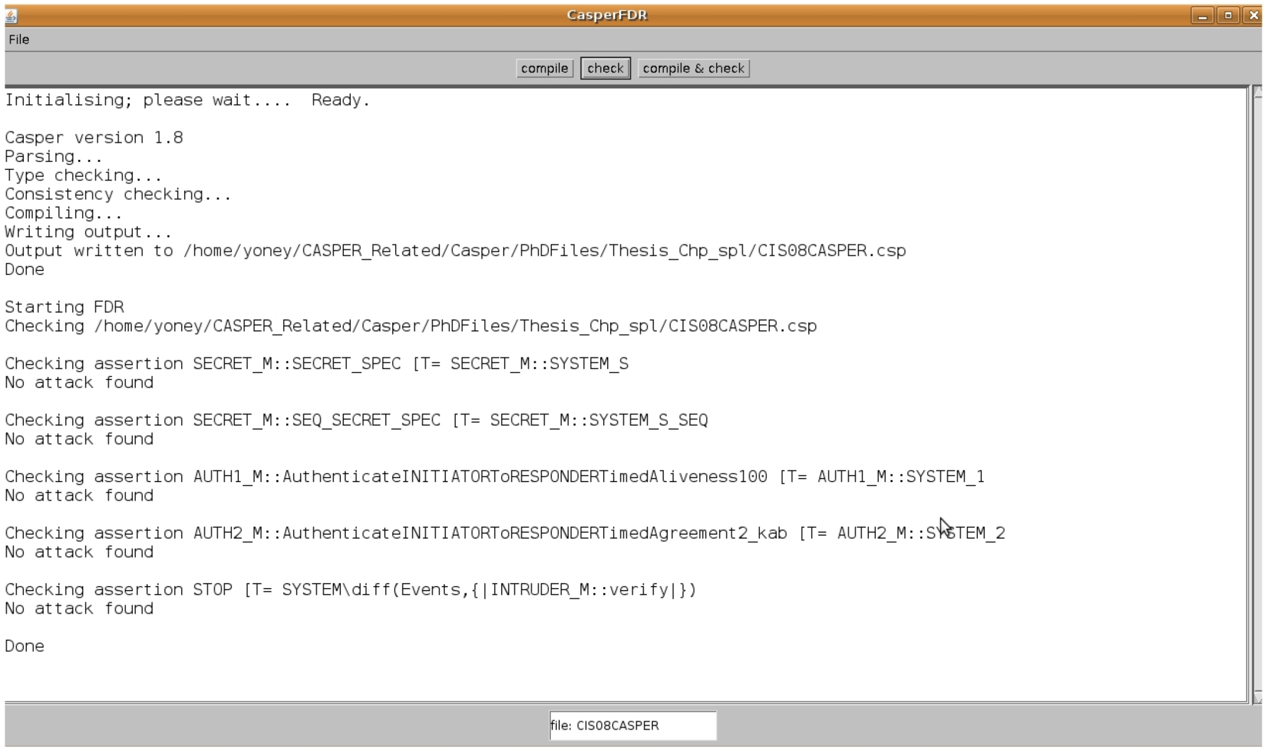
- `TimedAgreement(A, B, 2, [kab])` is a timed version of `Agreement(A,B,[])` specifies that *A* is correctly authenticated to *B*, and the agents agree upon **SessionKey**, *kab*, where *A*'s run is within the *2* **time units** of *B*.

- `TimedAliveness(A, B, 100)` mentions if *B* has successfully completed a run of the protocol with *A*, then *A* has previously been running the protocol within the *100* **time units**.

In the "#Intruder Information" section, an intruder's identity and the set of data values that knows, are mentioned. The *"Intruder knowledge"* section holds the identifiers and functions of the protocol that are known and can be applied to any other value such as "*Crackable, Guessable*" to those identifiers and functions. The capability of the attacker under assumptions are presented.

Figure 5.4 demonstrates the feasibility of the protocol and the achievements of the *intruder* with certain knowledge that is defined as `#Intruder Information` above CASPER script.

When the protocol is checked through FDR, due to the `StrongSecret(A, passwd(A,B), [B])`, `TimedAliveness(A, B, 100)` and `TimedAgreement(A,B,2,[kab])` specifications, there were *no attack found*. The ascertains are checked in Figure 5.5. The check is successful, where no possible attacks against the protocol have taken place. It has been observed that time to break the encryption (i.e. time to find flaw within the protocol) has significantly increased compared to the designed protocols in Sections 3.3, 4.2, 5.2 taken as references. The results are presented in Table 5.1.
The table shows that the protocol that is designed in this section, is better to previously reported ones in terms of *unit times* to find flaw within the protocol.



```
Initialising; please wait....  Ready.

Casper version 1.8
Parsing...
Type checking...
Consistency checking...
Compiling...
Writing output...
Output written to /home/yoney/CASPER_Related/Casper/PhDFiles/Thesis_Chp_spl/CIS08CASPER.csp
Done

Starting FDR
Checking /home/yoney/CASPER_Related/Casper/PhDFiles/Thesis_Chp_spl/CIS08CASPER.csp

Checking assertion SECRET_M::SECRET_SPEC [T= SECRET_M::SYSTEM_S
No attack found

Checking assertion SECRET_M::SEQ_SECRET_SPEC [T= SECRET_M::SYSTEM_S_SEQ
No attack found

Checking assertion AUTH1_M::AuthenticateINITIATORToRESPONDERTimedAliveness100 [T= AUTH1_M::SYSTEM_1
No attack found

Checking assertion AUTH2_M::AuthenticateINITIATORToRESPONDERTimedAgreement2_kab [T= AUTH2_M::SYSTEM_2
No attack found

Checking assertion STOP [T= SYSTEM\diff(Events,{|INTRUDER_M::verify|})
No attack found

Done
```

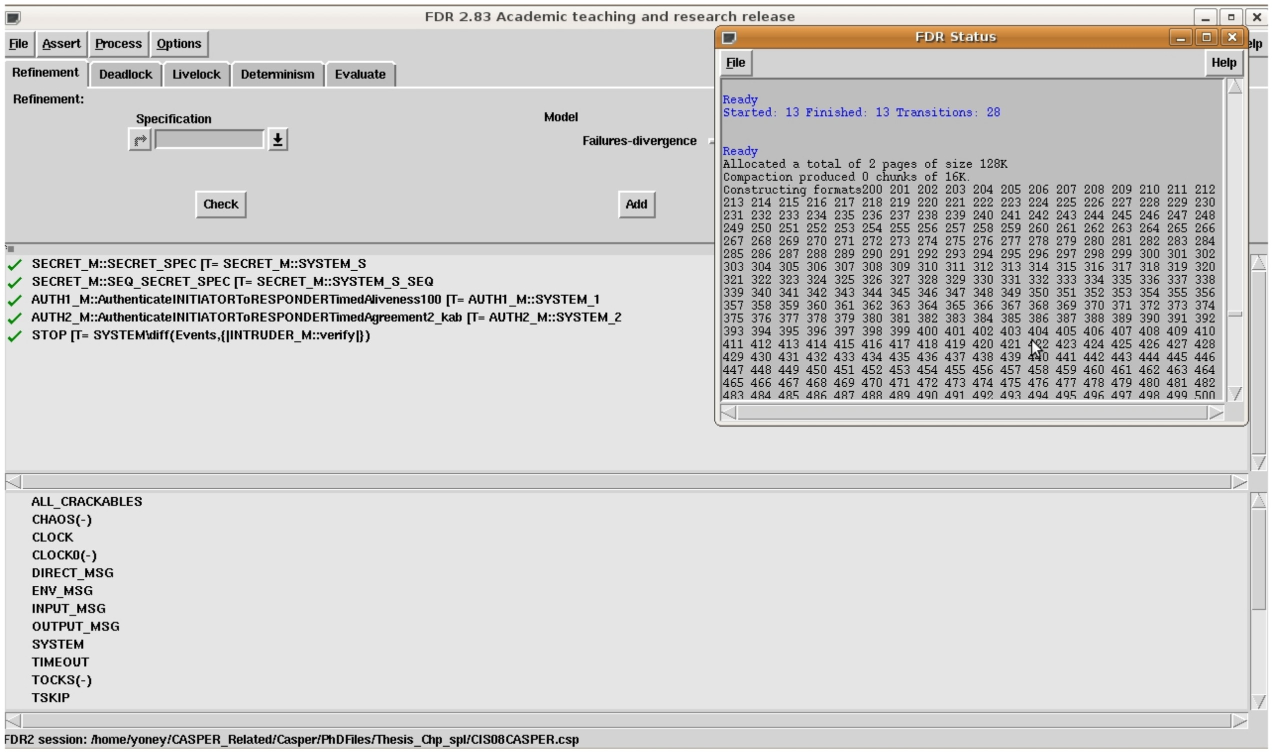
Figure 5.4: Analysis of further improvements to the timed authentication and frequent key renewal

5.4 Constructing the Rank Functions of Kerberos for the Combined use of the Timed Authentication Protocol and Frequent Key Renewal

This section presents a CSP model and construction of rank functions for the designed protocols that are mentioned in Sections 5.2 and 5.3 which shut down external access to Kerberos timed authentication protocol for a period of time, to enable the distribution of randomly generated keys in a relatively secure way. The CSP model and Rank functions are based on the extensive rules of [Schneider, 1998] and [Roscoe, 1995].

5.4.1 The CSP Model

In this section the CSP representation of the designed protocol of Section 5.3 is modelled as a network and specified the authentication property. As mentioned earlier in Section 4.3.2, while modelling the different processes of a protocol, advantage of the extensibility of CSP are used. The following scripts are representations of three *participants*, *INITIATOR,*



Refinement:

Specification

Model

Failures-divergence

```
✓ SECRET_M::SECRET_SPEC [T= SECRET_M::SYSTEM_S
✓ SECRET_M::SEQ_SECRET_SPEC [T= SECRET_M::SYSTEM_S_SEQ
✓ AUTH1_M::AuthenticateINITIATORToRESPONDERTimedAliveness100 [T= AUTH1_M::SYSTEM_1
✓ AUTH2_M::AuthenticateINITIATORToRESPONDERTimedAgreement2_kab [T= AUTH2_M::SYSTEM_2
✓ STOP [T= SYSTEM\diff(Events,{|INTRUDER_M::verify|})
```

```
Ready
Started: 13 Finished: 13 Transitions: 28


Ready
Allocated a total of 2 pages of size 128K
Compaction produced 0 chunks of 16K.
Constructing formats200 201 202 203 204 205 206 207 208 209 210 211 212
213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230
231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248
249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266
267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284
285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302
303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320
321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338
339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356
357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374
375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392
393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410
411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428
429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446
447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464
465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482
483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500
```

```
ALL_CRACKABLES
CHAOS(-)
CLOCK
CLOCK0(-)
DIRECT_MSG
ENV_MSG
INPUT_MSG
OUTPUT_MSG
SYSTEM
TIMEOUT
TOCKS(-)
TSKIP
```

FDR2 session: /home/yoney/CASPER_Related/Casper/PhDFiles/Thesis_Chp_spl/CIS08CASPER.csp

Figure 5.5: Analysis of specifications of further improvements to the timed authentication and frequent key renewal

RESPONDER, SERVER of the proposed protocol that are defined in above "#Processes" section in Section 5.2.

```
INITIATOR(A, S, na) =
    [] B : Agent @ A != B & env_I.A.(Env0, B,<B>) ->
    output.A.S.(Msg1, Encrypt.(SKey(A), <B>),<>) ->
    [] kab : SessionKey @ [] ts : TS @ [] pkb : addGarbage_(PublicKey) @
      input.S.A.(Msg2, Encrypt.(passwd(A, B), <S, A,
      Timestamp.ts, Encrypt.(PK(A), <kab>), pkb>),<>) ->
    [] ta : TS @
      output.A.B.(Msg3, Encrypt.(passwd(A, B), <A,
      Timestamp.ta, na, Encrypt.(pkb, <kab>)>),<>) ->
    [] nb : Nonce @ [] tb : TS @
      input.B.A.(Msg6, Encrypt.(inverse(kab), <nb, na, Timestamp.tb>),<>) ->
    output.A.B.(Msg7, Encrypt.(kab, <nb, Timestamp.ta>),<kab>) ->
    close.A.INITIATOR_role -> STOP

RESPONDER(B, S, nb) =
    [] A : Agent @
    [] v : addGarbage_({Encrypt.(passwd(A, B), <A, Timestamp.ta, na, Encrypt.(pkb, <kab>)>)
    | A <- Agent, B <- Agent, kab <- SessionKey,
```

Table 5.1: Comparison of times taking to break an encryption

Protocol	Time(secs.)
Framework without shut down and delay-decryption (Protocol in Section 3.3)	68
Designed protocol with delay-decryption (Protocol in Section 4.2)	110
Designed protocol with shut down and delay-decryption (Protocol in Section 5.2)	140
Designed protocol with delay-decryption and password encrypted (Protocol in Section 5.3)	360

```
    na <- Nonce, ta <- TS, pkb <- addGarbage_(PublicKey)}) @
      A != B & input.A.B.(Msg3, v,<>) ->
    [] tb : TS @
      output.B.S.(Msg4, Encrypt.(SKey(B), <A, Timestamp.tb>),<>) ->
    [] ts : TS @ [] pka : addGarbage_(PublicKey) @ [] now : TS @
      decryptable(v, pka) and nth(decrypt(v,pka), 1) == A
    and nth(decrypt(v,pka), 2) == now  and
    decryptable(nth(decrypt(v,pka), 3), passwd(A,B))
    and decryptable(nth(decrypt(v,pka), 4), SK(B)) &
      input.S.B.(Msg5, Encrypt.(passwd(A, B), <S, B, Timestamp.ts, pka>),<Timestamp.now>) ->
    RESPONDER'(B, S, nb, A, v, tb, ts, pka, nth(decrypt(nth(decrypt(v,pka),3))))
RESPONDER'(B, S, nb, A, v, tb, ts, pka, na) =
  RESPONDER''(B, S, nb, A, v, tb, ts, pka, na, nth(decrypt(nth(decrypt(v,pka),4),SK(B)),1))
RESPONDER''(B, S, nb, A, v, tb, ts, pka, na, kab) =
    output.B.A.(Msg6, Encrypt.(kab, <nb, na, Timestamp.tb>),<>) ->
    [] ta : TS @
      input.A.B.(Msg7, Encrypt.(inverse(kab), <nb, Timestamp.ta>),<kab>) ->
    close.B.RESPONDER_role -> STOP

SERVER(S, kab) =
    [] A : Agent @ [] B : Agent @
      input.A.S.(Msg1, Encrypt.(SKey(A), <B>),<>) ->
    [] ts : TS @
      output.S.A.(Msg2, Encrypt.(passwd(A, B), <S, A, Timestamp.ts, Encrypt.(PK(A), <kab>), PK(B)>),<>) ->
    [] tb : TS @
      input.B.S.(Msg4, Encrypt.(SKey(B), <A, Timestamp.tb>),<>) ->
    output.S.B.(Msg5, Encrypt.(passwd(A, B), <S, B, Timestamp.ts, PK(A)>),<>) ->
    close.S.SERVER_role -> STOP
```

The keywords *input* and *output* are used in the same way that are used earlier in Section 4.3.2.

The *temporary shut down link access* and *frequent key renew* and how they are designed within the protocol is explained under the "#Protocol description" in Sections 5.2 and 5.3. This is verified with the following CSP script:

```
 A != B & input.A.B.(Msg3, v,<>) ->
    [] tb : TS @
      output.B.S.(Msg4, Encrypt.(SKey(B), <A, Timestamp.tb>),<>) ->
    [] ts : TS @ [] pka : addGarbage_(PublicKey) @ [] now : TS @
```

Referring the test of *Message 5* "*decryptable*", "*decrypt*" and "*nth(,n)*" functions that are provided by CASPER are used in above Section 5.2, more assignments are needed in the *RESPONDER, B*:

```
input.S.B.(Msg5, Encrypt.(passwd(A, B), <S, B, Timestamp.ts, pka>),<Timestamp.now>) ->
RESPONDER'(B, S, nb, A, v, tb, ts, pka, nth(decrypt(nth(decrypt(v,pka),3))))
RESPONDER'(B, S, nb, A, v, tb, ts, pka, na) =
RESPONDER''(B, S, nb, A, v, tb, ts, pka, na, nth(decrypt(nth(decrypt(v,pka),4),SK(B)),1))
RESPONDER''(B, S, nb, A, v, tb, ts, pka, na, kab) =
output.B.A.(Msg6, Encrypt.(kab, <nb, na, Timestamp.tb>),<>) ->
 [] ta : TS @
```

The first assignment, `RESPONDER'(B,S,nb,A,v,tb,ts,pka, nth(decrypt(nth(decrypt(v,pka),3))))` assigns the **nonce** *na* as the **third** field of *Message 3*, but the message itself, is encrypted with the **password**, *passwd(A,B)*, has to be decrypted using the inverse of this key which is itself. The second assignment, `RESPONDER'(B,S,nb,A,v,tb,ts,pka,na)`= `RESPONDER''(B,S,nb,A,v,tb,ts,pka,na,nth(decrypt(nth(decrypt(v,pka),4),SK(B)),1))` assigns the **SessionKey**, *kab* as the **fourth** component of *Message 3* but the first field of the message is encrypted with B's **public key**, *PK(B)* and decryption has to be done by using the inverse of this key which is **secret key**, *SK(B)* of B.

Since the designed, proposed protocol is time sensitive (i.e timed authentication protocol), introduction of any delay will prevent the intruder's attempt to launch an attack. The following CSP process implies that *A* was running the protocol with *B* and there is mutual agreement between them. This mutual agreement depends on the value of **SessionKey**, *kab* within specified *unit time* for **Aliveness** chosen:

```
AuthenticateINITIATORToRESPONDERTimedAliveness100_0(A) =
    addTime(
      signal.Running1?A_role_!A?C_ ->
      CHAOS({signal.Commit1.RESPONDER_role.B.A | B <- Agent}),
      100)
AlphaAuthenticateINITIATORToRESPONDERTimedAliveness100_0(A) =
    union(
      {|signal.Running1.A_role_.A.B,
        signal.Commit1.RESPONDER_role.B.A |
           B <- inter(Agent, HONEST), A_role_ <- HONEST_ROLE|},
      {tock})
AuthenticateINITIATORToRESPONDERTimedAliveness100 =
    (AuthenticateINITIATORAliceToRESPONDERTimedAliveness100
    [| inter(AlphaAuthenticateINITIATORToRESPONDERTimedAliveness100_0(Alice),
             AlphaAuthenticateINITIATORToRESPONDERTimedAliveness100_0(Bob)) |]
    AuthenticateINITIATORBobToRESPONDERTimedAliveness100)

AuthenticateINITIATORToRESPONDERTimedAgreement2_kab_0(A) =
    addTime(
      signal.Running2.INITIATOR_role.A?B?kab ->
      signal.Commit2.RESPONDER_role.B.A.kab -> STOP,2)
AlphaAuthenticateINITIATORToRESPONDERTimedAgreement2_kab_0(A) =
    union(
      {|signal.Running2.INITIATOR_role.A.B,
        signal.Commit2.RESPONDER_role.B.A |
           B <- inter(Agent, HONEST)|},
      {tock})
AuthenticateINITIATORToRESPONDERTimedAgreement2_kab =
    (AuthenticateINITIATORAliceToRESPONDERTimedAgreement2_kab
    [| inter(AlphaAuthenticateINITIATORToRESPONDERTimedAgreement2_kab_0(Alice),
             AlphaAuthenticateINITIATORToRESPONDERTimedAgreement2_kab_0(Bob)) |]
    AuthenticateINITIATORBobToRESPONDERTimedAgreement2_kab)
```

In addition to the *nonces*, *timestamps* are also being used for time agreement between the processes A and B. Relating the above CSP codes of the `TimedAliveness(A,B,100)` and `TimedAgreement(A,B,2,[kab])`, the system is verified :

```
SYSTEM_1 =
    let Agent_renamed_ = ALGEBRA_M::applyRenamingToSet(Agent)
        SessionKey_renamed_ = ALGEBRA_M::applyRenamingToSet(SessionKey)
        Nonce_renamed_ = ALGEBRA_M::applyRenamingToSet(Nonce)
        TS_renamed_ = ALGEBRA_M::applyRenamingToSet(TS)
    within
    SYSTEM
      [[send.A.B.ALGEBRA_M::rmb((Msg7, Encrypt.(kab, <nb, Timestamp.ta>), <kab>)) <-
          signal.Running1.INITIATOR_role.A.B,
        receive.A.B.ALGEBRA_M::rmb((Msg7, Encrypt.(kab, <nb, Timestamp.ta>), <kab>)) <-
          signal.Commit1.RESPONDER_role.B.A |
            A <- Agent_renamed_, B <- Agent_renamed_,
            kab <- SessionKey_renamed_, nb <- Nonce_renamed_,
            ta <- TS_renamed_
      ]]
   \ diff(Events, alphaAuthenticateINITIATORToRESPONDERTimedAliveness100)

SYSTEM_2 =
    let Agent_renamed_ = ALGEBRA_M::applyRenamingToSet(Agent)
        SessionKey_renamed_ = ALGEBRA_M::applyRenamingToSet(SessionKey)
        Nonce_renamed_ = ALGEBRA_M::applyRenamingToSet(Nonce)
        TS_renamed_ = ALGEBRA_M::applyRenamingToSet(TS)
    within
    SYSTEM
      [[send.A.B.ALGEBRA_M::rmb((Msg7, Encrypt.(kab, <nb, Timestamp.ta>), <kab>)) <-
          signal.Running2.INITIATOR_role.A.B.kab,
        receive.A.B.ALGEBRA_M::rmb((Msg7, Encrypt.(kab, <nb, Timestamp.ta>), <kab>)) <-
          signal.Commit2.RESPONDER_role.B.A.kab |
            A <- Agent_renamed_, B <- Agent_renamed_,
            kab <- SessionKey_renamed_, nb <- Nonce_renamed_,
            ta <- TS_renamed_
      ]]
   \ diff(Events, alphaAuthenticateINITIATORToRESPONDERTimedAgreement2_kab)
```

The following CSP code is defined to represent the "#Intruder Information" and *Intruder-Knowledge* definitions with its relevance to the initial knowledge, where detailed explanations are given in Sections 4.2 and 5.2:

```
IK0_init = union({Alice, Bob, Mallory, Sam, Nm, SK(Mallory), SKey(Mallory),
            Garbage}, TimeStamp)
INTRUDER_1 =
    (chase(INTRUDER_0)
      [[ hear.m_ <- send.A_.B_.(l_,m_,se_) |
           (l_,m_,se_,re_) <- DIRECT_MSG,
           A_ <- diff(SenderType(l_),{Mallory}), B_ <- ReceiverType(l_) ]]
     [|{| hear |}|] STOP)
      [[ say.m_ <- receive.A_.B_.(l_,m_,re_) |
           (l_,m_,se_,re_) <- DIRECT_MSG,
           A_ <- SenderType(l_), B_ <- ReceiverType(l_) ]]
```

The keywords *hear, say* are used to represent hearing and saying a message during the authentication and transfer of a message across network. `(l_,m_,se_,re_)` is types of *messages* that are *sent* and *received* by processes (agents) as they are considered by those agents.

Addition to the *intruder* definition, the initially known facts are added under the following CSP script:

```
SAY_KNOWN_0 =
    (inter(IK1, ALL_SECRETS_DI) != {} & dummy_leak -> SAY_KNOWN_0)
    [] dummy_send -> SAY_KNOWN_0
    [] dummy_receive -> SAY_KNOWN_0
 SAY_KNOWN =
    SAY_KNOWN_0
      [[ dummy_leak <- leak.f_ | f_ <- inter(IK1, ALL_SECRETS_DI) ]]
      [[ dummy_send <- send.A_.B_.(l_,m_,se_) |
           (l_,m_,se_,re_) <- DIRECT_MSG, components_(m_) <= IK1,
           A_ <- diff(SenderType(l_),{Mallory}), B_ <- ReceiverType(l_) ]]
      [[ dummy_receive <- receive.A_.B_.(l_,m_,re_) |
           (l_,m_,se_,re_) <- DIRECT_MSG, components_(m_) <= IK1,
           A_ <- SenderType(l_), B_ <- ReceiverType(l_) ]]
  STOP_SET = {| send.Mallory |}

INTRUDER =
    INTRUDER_1 [| STOP_SET |] STOP

IntruderInterface = Union({{| send, receive |}, {|crack|}})
AlphaSystem = {|env, send, receive, close, tock|}
SystemManagerInterface = inter(AlphaSystem,CRACKING_M::AlphaManager)
SYSTEM = (SYSTEM_M::TSYSTEM_1 [|SystemManagerInterface|] CRACKING_M::Manager)
           [|IntruderInterface|] INTRUDER_M::INTRUDER
```

The keyword *leak* is used to signal that a possible secret has been learnt. Elements such as `f_` , `IK1` are components of the *intruder* for currently unkown *fact* and *initial knowledge.* Despite the highly specified *intruder* and its connection attempts, the designed protocol is successful.

5.4.2 The Rank Functions

In this section the rank functions which are listed in Section 2.3.1, of the **NET** on the message space are constructed. The following steps and Table 5.2 show the rules that are constructed while creating rank functions.

1. All user **IDs** in the set $\boldsymbol{U}$ are assigned to a *positive rank*: It is assumed that **IDs** are known to *intruder* and can be impersonated.

2. All the nonces in $\boldsymbol{N}$ are assigned to a *positive rank*: It is also assumed that **Nonces** are known to *intruder Knowledge.*

3. In the protocol there are three different keys: **SessionKey** which is defined as *kab*, **ServerKey** as *SKey* and **Password** as *passwd. kab, passwd* are assigned to a *non-positive rank*, because they are supposed to be private to agents in the protocol. However, *SKey* is assigned to a *positive rank*, with the same rule that is mentioned in previous two steps.

4. The running messages between *input* and *output* channels are assigned to a *non-positive rank.*

5. This step is connected with the previous step which is related with signal events *Commit*, and *Running* where represented as *input* and *output* channels respectively.

Table 5.2: Rank functions for the protocol

Step	Rank Function
1	$\rho(\boldsymbol{U}) = 1$
2	$\rho(\boldsymbol{N}) = 1$
3	$\rho(\boldsymbol{K}) = \{$ 0 if: kab or passwd else: 1 (SKey)
4	$\rho(m_K) = \{$ 0 if: m(kab) or m(passwd) else:1 (m(SKey))
5	ρ(A,B,nb,na) = 1 and ρ(B,A,kab, na) = 0

The rank function theorem is defined in terms of general sets $\boldsymbol{R}$ and $\boldsymbol{T}$. In this research, $\boldsymbol{R}$ and $\boldsymbol{T}$ are assigned to step 5 of above Table 5.2 and it is extended as:

$$Initiator - output = \boldsymbol{R} = \rho(\boldsymbol{Running}, \text{A,B}, nb,\ na) = 1$$
$$Initiator - input = \boldsymbol{T} = \rho(\boldsymbol{Commit}, \text{B,A}, kab,\ na) = 0$$

The constructed rank functions (as provided above Table 5.2) are employed with the conditions of the rank function theorem, [Schneider, 1998] (discussed in Section 2.3.1), in order to make sure that the functions provided satisfy the conditions of the theorem.

1. $\forall\ m \in \mathbf{IK} \bullet \rho(m) > 0$
   In protocol description **IK** is *Intruder Knowledge* and at the beginning it contains all users in the **NET** and *nonce* of itself, where all the contents are positive rank. Therefore, the set is *positive rank*. The condition is deemed satisfied.

2. $\forall\ \mathbf{S} \subseteq M,\ m \in M \bullet ((\forall\ m' \in \mathbf{S} \bullet \rho(m') > 0) \bigwedge \mathbf{S} \vdash m) \Rightarrow \rho(m) > 0$
   This condition checks whether a message of a *non-positive rank* can be generated from a given set of messages of *positive rank*. Messages are non-positive rank (step 4 of Table 5.2). Intruder generates any messages that are *non-positive rank*. These messages are messages of steps 3, 5 and 6 at protocol description, which are generated by *passwd,kab*. Both are *non-positive rank*. This prevents intruder from generating these keys. Therefore there is no way of getting the messages. This condition is satisfied.

3. $\forall\ t \in \mathbf{T} \bullet \rho(t) \leq 0$

   In this condition, all events in T are non-positive rank. Since $Initiator - input = \rho(\text{B,A,}kab,\ na) = \rho(\boldsymbol{Commit},\text{B,A,}kab,\ na)$ is *non-positive rank.* This condition is satisfied.

4. $\forall\ i \in \boldsymbol{U} \bullet \text{User}_i \underset{R}{\parallel} \text{Stop}$ **maintains** ρ

   Every process in **NET** needs to maintain positive ρ. However, events are <u>*restricted*</u> in set R. $Initiator - output = \rho(\text{A,B,}nb,na) = 1 = \boldsymbol{R}$. A and B are restricted on $Initiator - output$. Therefore it will be checked if they maintain positive ρ.

$$\begin{array}{lll} \text{User}_A \parallel \text{Stop} = & \Box_b & output.A.B.(Msg3, Encrypt.(passwd(A,B), < \\ Initiator - output(A,B,nb,na) & & A, Timestamp.ts, na, Encrypt.(pkb, < kab >) >,) \\ & & <>)- > []nb : Nonce@[]tb : TS@input.B.A.(Msg6, \\ & & Encrypt.(inverse(kab), < nb, na, Timestamp.tb >), \\ & & <>)- > []ta : TS@output.A.B.(Msg7, Encrypt.(kab, \\ & & < nb, Timestamp.ta >), < kab >)- > STOP \end{array}$$

   $\Box_b$ is choice operation and b indicates the other participants (B and S). $b = B$ where communicates with $User_A$. In terms of protocol run, $User_A \underset{Initiator-output(A,B,nb,na)}{\parallel} Stop$ does not fail to maintain positive ρ, on call of *output.A.B* with *na* and *nb* in *Message 3* and *Message 7* respectively.

The protocol is checked and proven to be successful in improving security against attacks despite strong intruder connection attempts. This was demostrated in steps given in this section.

5.5 Conclusion

This chapter is concerned with improving the security of Kerberos authentication protocol. A new approach and a protocol are derived from the designed and verified protocols in Sections 3.3, and 4.2, to increase time for an intruder to break an encryption and hence improve the security. Security is further improved by restricting externals' access to the system and during this time, distribution of the frequently renewed keys and encryption/decryption of messages are controlled with timed authentication. The availability of the protocols are checked with CASPER. The capability of the attacker under assumptions are presented with inductive capability of FDR and results are compared.

This chapter presented the CSP codes and the construction of rank functions of the designed protocol in Section 5.3. CSP processes and rank functions that are constructed, give opportunities of better understanding of security protocols and focuses on relevant design aspects of these protocols. The new protocol is an enhanced version of previously reported protocols in Sections 4.2 and 5.2.

Schneider's CSP processes and rank function theorem [Schneider, 1998] are applied to expose any flaws in the design and possible attacks have been successfully identified. Analyses show that the protocol developed has achieved the goal of increasing the time needed to break the encryption, and hence improve security.

Since the proposed approach involves temporary interruption to link/server access, it has implications in terms of performance degradation. Analytical methods will be used in the following chapter to evaluate the cost in terms of the degradation of system performance. The models being developed will consider the system for exact performability evaluation.

Chapter 6

Performability Modelling of a Kerberos Server with Frequent Key Renewal under Pseudo-Secure Conditions with Server Breakdowns and Repairs

6.1 Introduction

Despite the design and proposal of authentication approaches for improving security, threats of penetration and various other forms of attacks have continued to evolve. As discussed in Section 2.4, the protection of data, system and resources presents even more challenging tasks for scientists and network engineers; while systems are configured to ensure reliable, secure communications, privacy, and data integrity, the performance of the underlying networks should also be taken into account. In other words, the systems considered should still be able to perform sufficiently and legitimate access to resources should not be compromised while the security policies are applied.

In this chapter, performance evaluation techniques are employed for security protocols that are proposed and modelled previously. Analytical models are developed in order to analyse the effects of frequent key renewal under pseudo-secure conditions of Section 5.2, and to evaluate the cost in terms of the degradation of system performance with random failures of the authentication server in Sections 6.2 and 6.3 respectively. Numerical results are presented and discussed.

6.2 Performability Modelling with Frequent Key Renewal

Since the proposed model is based on secure key distributions at various intervals, during key distribution, external access to the network incorporating authentication servers is not allowed. The access restriction applies for short intervals, however, any link shut down costs the network in terms of performance degradation. Therefore, it is essential to evaluate the impact of the proposed approach on system performance.

For this purpose, an analytical model has been developed to evaluate the performability of the proposed approach. While key distribution times depend on network characteristics such as size, speed, congestion etc., the frequency of key renewals can be determined by the mean values of decryption times.

6.2.1 The Model

This analytical model considered is a simple K-server system with an unbounded queue, Poisson arrivals and exponentially distributed service times. Jobs arrive at the system in a Poisson stream at a rate σ, and join the queue. Jobs are homogeneous and the service times of jobs serviced by server are distributed exponentially with mean $1/\mu$. The distribution of time intervals between shut downs are exponential and given by mean $1/\delta$. Once the system is shut, the server does not provide service to incoming requests for an exponentially distributed duration (key distribution time) which is given by $1/\varphi$. Figure 6.1 shows the model of the system where, Figure 6.2 is the Markov chain that represents the states of the system.
When Figure 6.2 is considered, (0) represents the state where the server is shut to the external access since the key distribution takes place. In state (1) the server is operative since the system is not shut.

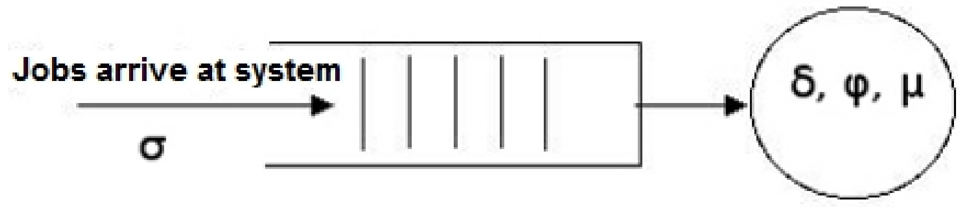


Figure 6.1: The model of Kerberos server

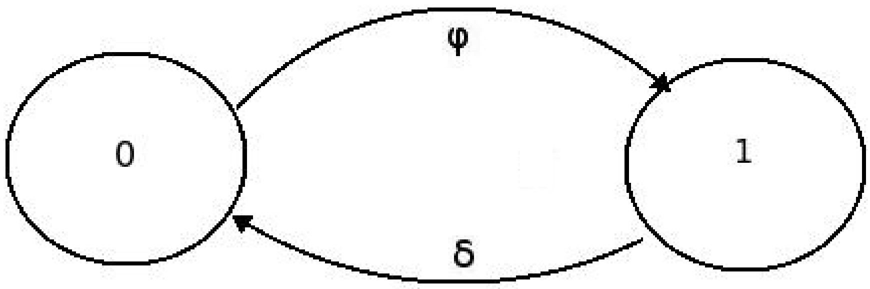


Figure 6.2: The states of Kerberos server

6.2.2 Two Dimensional Markov Representation of the System

The state of the system at time t can be described by a pair of integer valued random variables, *I(t)* and *J(t)*, specifying the operative state of the system and the number of jobs present, respectively. *I(t)*, $t \geq 0$, is an irreducible Markov process. *J(t)* is the total number of jobs in the system at time t, including the *one(s)* in service. Then, $Z = [I(t),J(t)];\ t \geq 0$ is an irreducible Markov process on a lattice strip (a QBD process), that models the system is shown in Figure 6.3. Similar studies are considered in [Chakka and Mitrani, 1994] and [Mitrani, 2005] for some general multi-server systems with single repairman *(R=1)*, for some repair strategies, however, protocols which cause transitions to make the system inoperative (such as key renewal), are not considered.
Here, A is the matrix of instantaneous transition rates from state *(i, j)* to state *(k, j)* (where $i \neq k$) with *zeros* on the main diagonal. These are the purely lateral transitions of the model Z. Matrices B and C are transition matrices for one-step upward and one-step downward transitions respectively. The transition rate matrices do not depend on j for $j \geq M$, where M is a threshold having an integer value. In this study the M value is taken as *one*. The process Z evolves with the following instantaneous transitions:

- $A_j(i,\ k)$: Purely lateral transition rate, from state *(i,j)* to state *(k, j)*, $(i=0,\ 1,\ \ldots,\ N;\ k=0,\ 1,\ \ldots,\ N;\ i \neq k;\ j=0,\ 1,\ \ldots,)$, caused by a change in the state (i.e., a break down, or shut down for key distribution).

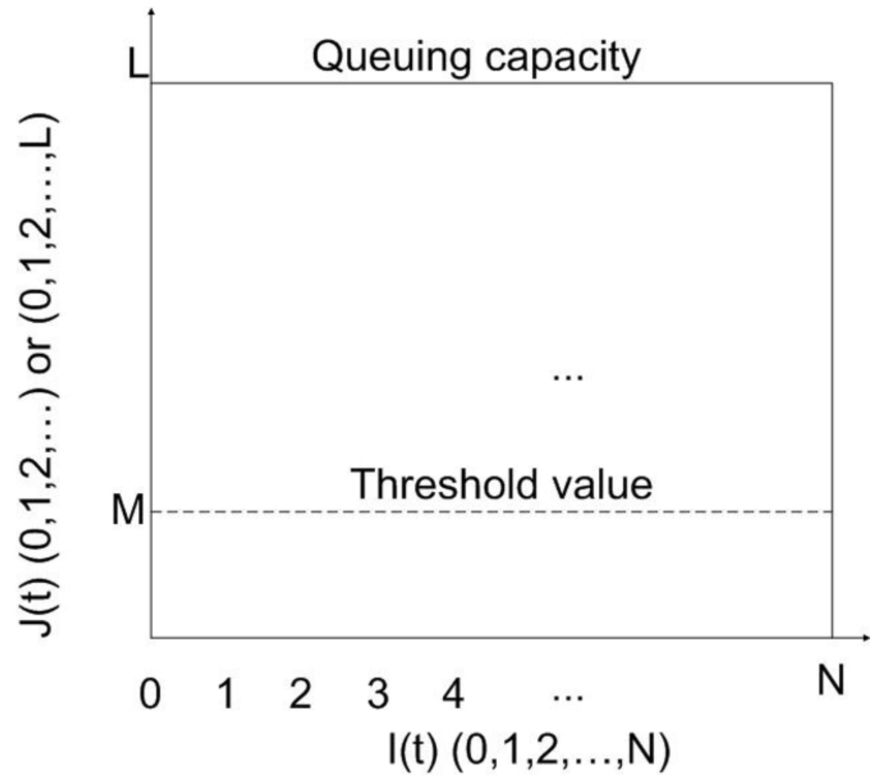


Figure 6.3: Two dimensional lattice strip

- B_j*(i, k)*: One-step upward transition rate, from state *(i,j)* to state *(k, j+1)*, *(i=0, 1, ..., N; k=0, 1, ..., N; and j=0, 1, ...)*, caused by a job arrival into the queue.
- C_j*(i, k)*: One-step downward transition rate, from state *(i,j)* to state *(k, j-1)*, *(i=0, 1, ..., N; k=0, 1, ..., N; and j=0, 1, ...)*, caused by the departure of a serviced job.

The total number of states for the Kerberos server with KDC subjected to failures is given by 2. Numbering these operative states can be arbitrary. We assign numbers (0, 1) to the states (0), (1) respectively. For a Kerberos server, with Frequent Key Renewal protocol A, B and C can be given as follows:

$$A = A_j = \begin{pmatrix} 0 & \varphi \\ \delta & 0 \end{pmatrix} \qquad B = B_j = \begin{pmatrix} \sigma & 0 \\ 0 & \sigma \end{pmatrix}$$

$$C = C_j = \begin{pmatrix} 0 & 0 \\ 0 & \mu \end{pmatrix}$$

6.2.3 Analytical Model's Numerical Results

Mean Queue Length (MQL) are computed as a function of the **mean arrival rate**, σ and various δ and φ values for a K-server system. In Figure 6.4, the MQL is shown for systems with various δ values. The other parameters are μ=200 per second and $1/\varphi$=10 seconds. The Figure shows the MQL performance of the systems with various intervals between shut downs.

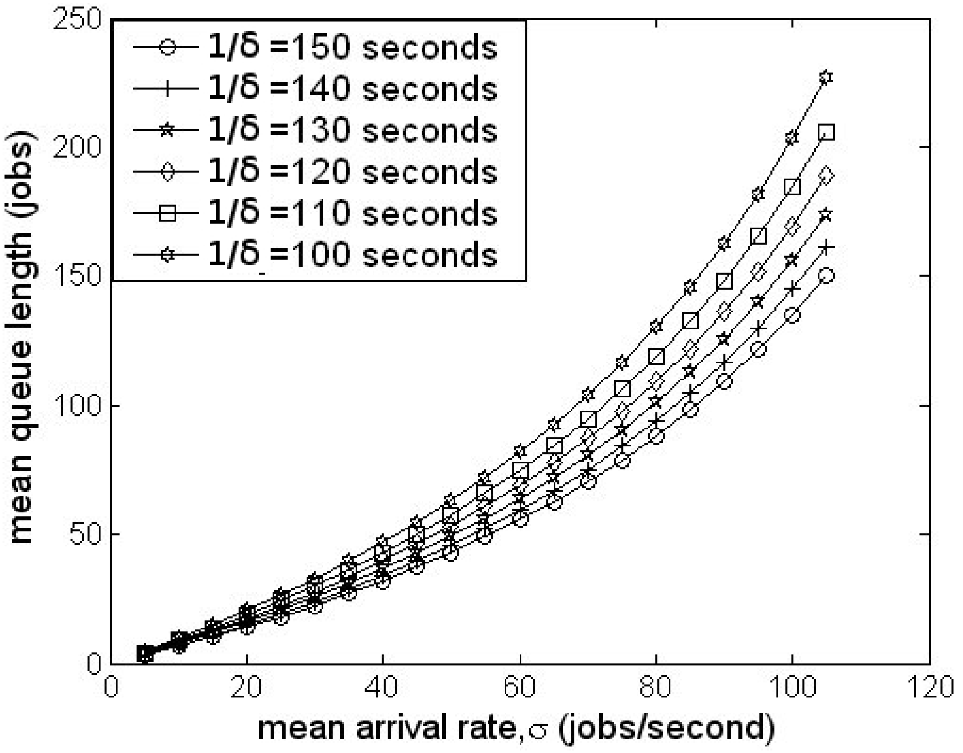


Figure 6.4: MQL as a function of σ for various δ values

To show the effects of various shut down durations, computations are performed for various φ values where $1/\delta$ is 140 seconds, and μ=200. The results are shown in Figure 6.5. As it can be seen from the Figure, as φ increases (the shut down duration decreases) the MQL increases significantly. Also in this Figure the MQL values of a system **without** shut downs is shown for comparison. As expected the system **without** shut downs performs better. Then, performance degradation increases gradually from negligible to significant. However, the Figure shows that especially for light traffic (lower arrival rates) it can be affordable to introduce shut downs in order to increase the degree of security.

Finally, in Figure 6.6, MQL is computed for systems with K-servers. Please note that the servers considered work in parallel and once the shut down is introduced they become non-

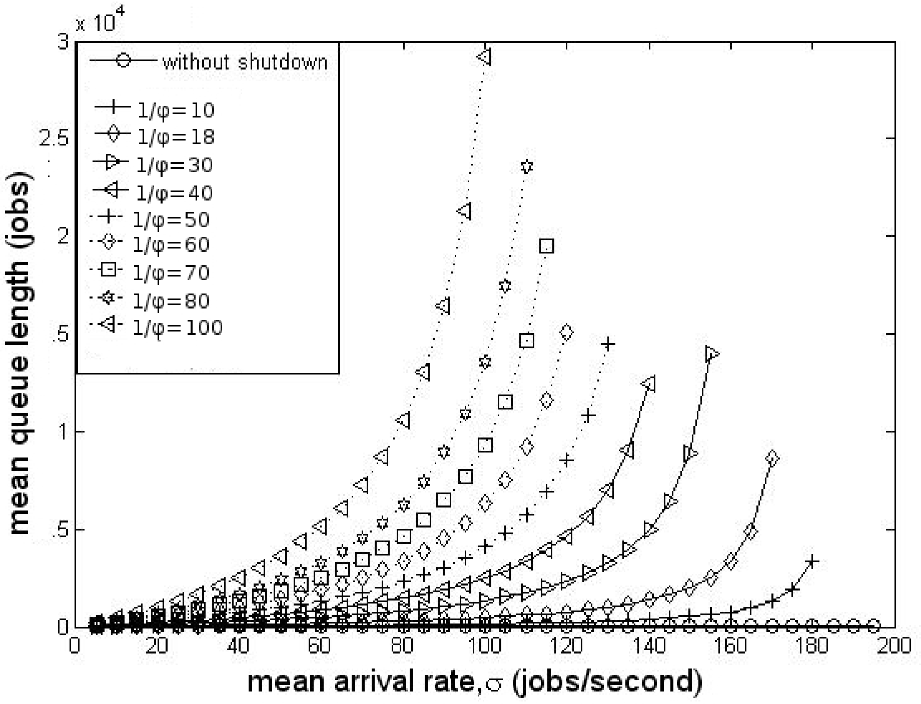


Figure 6.5: MQL as a function of σ for various φ values

operative for the same duration given by $1/\varphi$=10 seconds. Other parameters are given as $1/\delta$ =140 seconds and μ=200 jobs/second.

Figure 6.6 shows the contribution of an increase in K to the performance of the system with shut downs. However, since all of the servers needed to be shut down at the same time intervals for the same duration, the system's performance does not become better than the system without shut downs. However, the results show that it is possible to increase the degree of security while the systems performance is reasonable. Again, after a certain K value, increasing the number of servers further will not improve the MQL performance.

6.2.4 Discussions

Tests carried out for the proposed protocol(Section 5.2) and the network with shut downs revealed the following: The proposed protocol increases the time taken to break an encryption for about 25-75% compared to various other protocols, given in Table 5.1 and Section 2.4.2. However, it is possible to stop intruders' access during key distribution further increasing network security. For lightly loaded networks, performance degradation is small. For loaded networks, the performance degradation can be minimised by increasing the number of servers working in parallel which again comes at a cost. It must be noted that, after a certain

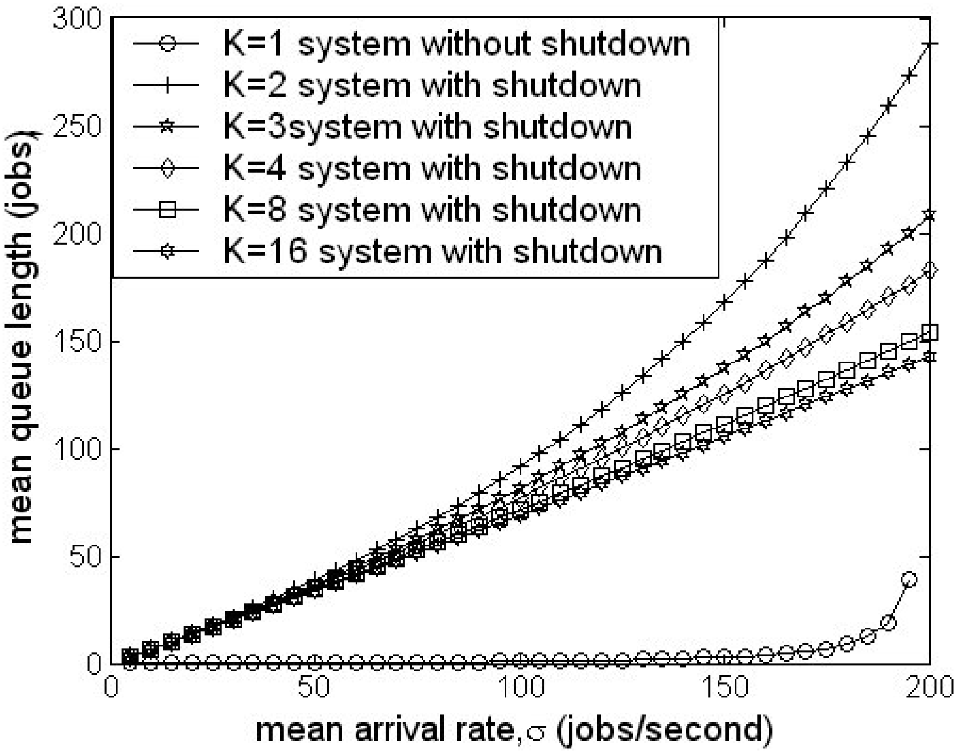


Figure 6.6: MQL for constant φ, δ and K-servers

K value, an increase in the number of servers will not have a significant effect on system performance. Optimum values for δ, and K can be chosen for maximising system performance for a pre-specified level of security. Metrics can be developed to determine the degree of security.

6.3 Performability Modelling with Breakdowns and Repairs

This section presents a model for a single Kerberos server suffering from potential KDC failures and used together with the Frequent Key Renewal protocol, designed in Section 5.3. Unlike the previous studies, the server failures [Brennen, 2004] are also considered together with the interruptions for key distributions. The section shows the two dimensional Markov model of a Kerberos server considered and the steady state solution approach. Then, numerical results for the performability measures are presented to show the effects of key renewals as well as server failures.

6.3.1 The Model

Allocation of jobs is usually done considering the availability of a Kerberos server. The values and the probabilistic distributions of the parameters used to develop the analytical models are mainly taken from [Mitrani, 2005] and from Section 6.2. The values of mean arrival and service rates are application dependent.

Kerberos server considered is a single server system with a queue which is unbounded since all of the incoming service requests are accepted. Jobs arrive at the system in a Poisson stream at a mean rate of σ. The service times of jobs are distributed exponentially with mean $1/\mu$. However, the Kerberos server considered can execute jobs only during its operative periods (during an operative period the processor is capable of its intended operation, whether working or idle). The Kerberos server is operative if it is not broken, or if the system is not shut. The periods where the Kerberos server is not broken is distributed exponentially with mean $1/\xi$. At the end of this period, the server breaks down and requires an exponentially distributed repair time with mean $1/\eta$. The distribution of time intervals between shut downs are exponential and given with mean $1/\delta$. When the system is shut, the server does not provide service to incoming request for an exponentially distributed duration (key distribution time) which is given by $1/\varphi$. Figure 6.7 is the Markov chain that represents the operative states of the system.

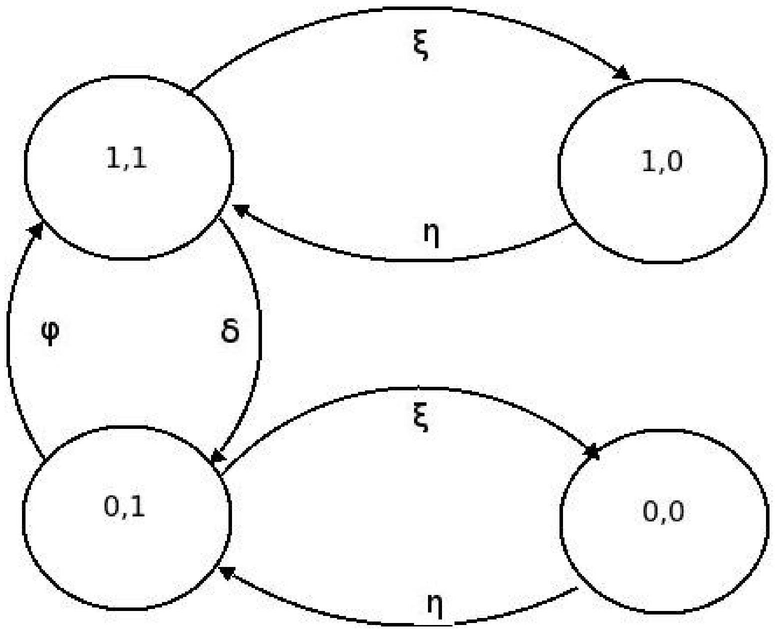


Figure 6.7: The operative states of Kerberos server

When Figure 6.7 is considered, (0,0) represents the state where the server is broken and the system is shut. In state (1,0) the system is not shut but the server is broken. In the state (0,1) the server is not broken but the system is shut since the key distribution takes place. Finally the state labelled as (1,1) is the state where the server is operative since the system is not shut and the server is not broken. Figure 6.7 clearly shows that there are no transitions between states (1,0) and (0,0), since system shut downs are not required when the server is broken.

Referring to the Section 6.2.2, the total number of states for the Kerberos server with KDC subjected to failures is given by 4. Numbering these operative states can be arbitrary. We assign numbers (0, 1, 2, 3) to the states (0,0), (0,1), (1,1), (1,0) respectively. For a Kerberos server, with Frequent Key Renewal protocol matrices A, B and C can be given as follows:

$$A = A_j = \begin{pmatrix} 0 & \eta & 0 & 0 \\ \xi & 0 & \varphi & 0 \\ 0 & \delta & 0 & \xi \\ 0 & 0 & \eta & 0 \end{pmatrix} \qquad B = B_j = \begin{pmatrix} \sigma & 0 & 0 & 0 \\ 0 & \sigma & 0 & 0 \\ 0 & 0 & \sigma & 0 \\ 0 & 0 & 0 & \sigma \end{pmatrix}$$

$$C = C_j = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

6.3.2 The Steady State Solution

The solution is given for an unbounded queue. The model developed has been solved using the spectral expansion solution [Mitrani, 2005]. The steady state probabilities are defined in rows as $\mathbf{v}_j = (p_{0,j}, p_{1,j}, ..., p_{N,j},)$, j=0, 1, 2, ... Defining diagonal matrices as:

- $\mathbf{D_j^A}$(i,i) $= \sum_{k=0}^{N} A_j(i,k)$
- $\mathbf{D_j^B}$(i,i) $= \sum_{k=0}^{N} B_j(i,k)$
- $\mathbf{D_j^C}$(i,i) $= \sum_{k=0}^{N} C_j(i,k)$

The steady state balance equations can then be written as follows:

- $\mathbf{v_0[D_0^A + D_0^B] = v_0A_0 + v_1C_1}$

- $\mathbf{v_j}[\mathbf{D_j^A} + \mathbf{D_j^B} + \mathbf{D_j^C}] = \mathbf{v_{j-1}}\mathbf{B_{j-1}} + \mathbf{v_j}\mathbf{A_j} + \mathbf{v_{j+1}}\mathbf{C_{j+1}}$; $1 \leq \mathrm{j} \leq$ M-1
- $\mathbf{v_j}[\mathbf{D^A} + \mathbf{D^B} + \mathbf{D^C}] = \mathbf{v_{j-1}}\mathbf{B} + \mathbf{v_j}\mathbf{A} + \mathbf{v_{j+1}}\mathbf{C}$; $\mathrm{j} \geq$ M-1

It is possible to solve the balance equations using the spectral expansion method similar to studies in [Chakka and Mitrani, 1994], [Haverkort and Ost, 1997] and [Mitrani, 2005]. Once the state probabilities are obtained, the MQL, can be computed as:

$$\mathrm{MQL} = \sum_{j=0}^{L} j \sum_{i=0}^{N} P(i,j)$$

6.3.3 Numerical Results and Discussions

The analytical model considered assumes an unbounded queue since incoming requests are not blocked. To show the effectiveness of the method presented, and to evaluate the performance of a Kerberos server with *frequent key renewal under pseudo-secure conditions*, numerical results are provided in this section. Effects of *key renewal* periods are analysed as well as the effects of KDC failures.

Figure 6.8 shows the MQL as a function of φ for various δ values. The other parameters are $1/\eta$=2 hours, $1/\xi$= 1000 hours, σ=80 jobs/sec, and μ=200 jobs/sec. It is clear that as key distribution time increases, the mean queue length also increases. Also, effect of increasing the key renewal period $1/\varphi$ decrease for greater $1/\delta$ values. This is mainly because other factors such as failures and repairs affect the system more significantly.

Figure 6.9 shows the probability of being in state (1,1) ($q_{(1,1)}$) as a function of φ for various δ. In state (1,1) the Kerberos server is not interrupted for key renewal and also it does not suffer from failures. (1,1) is the only state where the Kerberos server is operative. The other parameters are $1/\eta$=2 hours, $1/\xi$= 1000 hours, σ=80 jobs/sec, and μ=200 jobs/sec. Figure 6.9 shows that the probability of being in this operative state decreases significantly as the renewal time and time between renewals increases.

Effects of various failure rates are shown together with relatively high key renewal times in Figure 6.10a. The parameters are σ jobs/sec, μ=200 jobs/sec $1/\eta$=2 hours, $1/\varphi$= 70 seconds, and $1/\delta$=140 seconds. The figure clearly shows that failures affect the performance significantly. The same parameters are used for Figure 6.10b for $1/\varphi$= 10 seconds. The results given for the system with shorter key renewal times shows that the system performs significantly better than the one considered in Figure 6.10a.

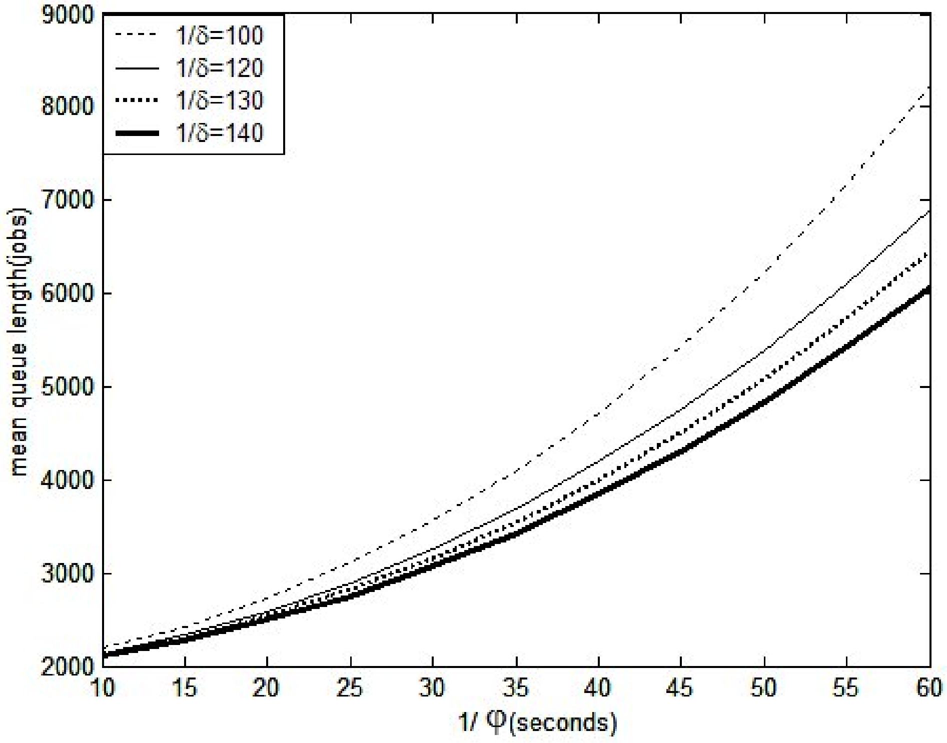


Figure 6.8: The MQL as a function of φ for various δ values

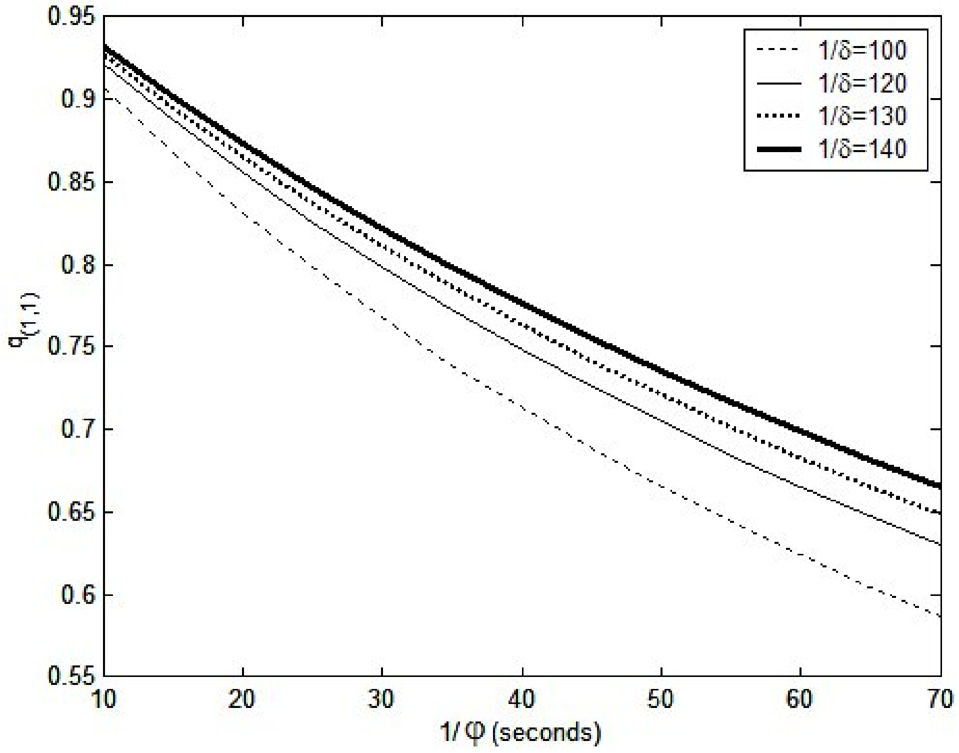
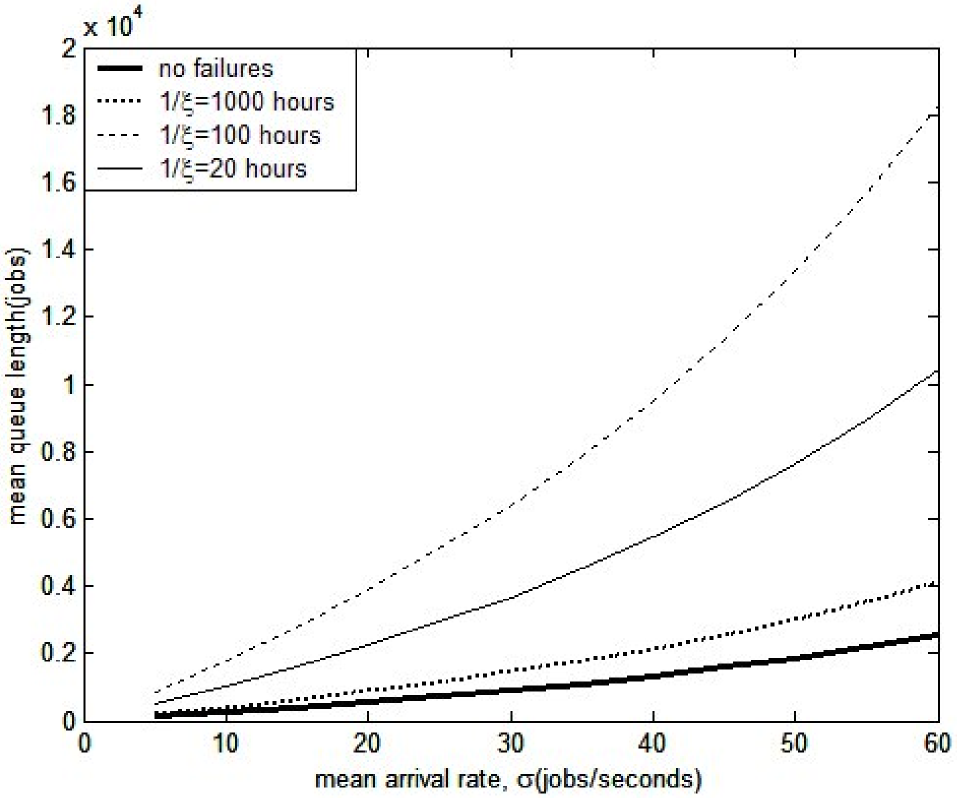
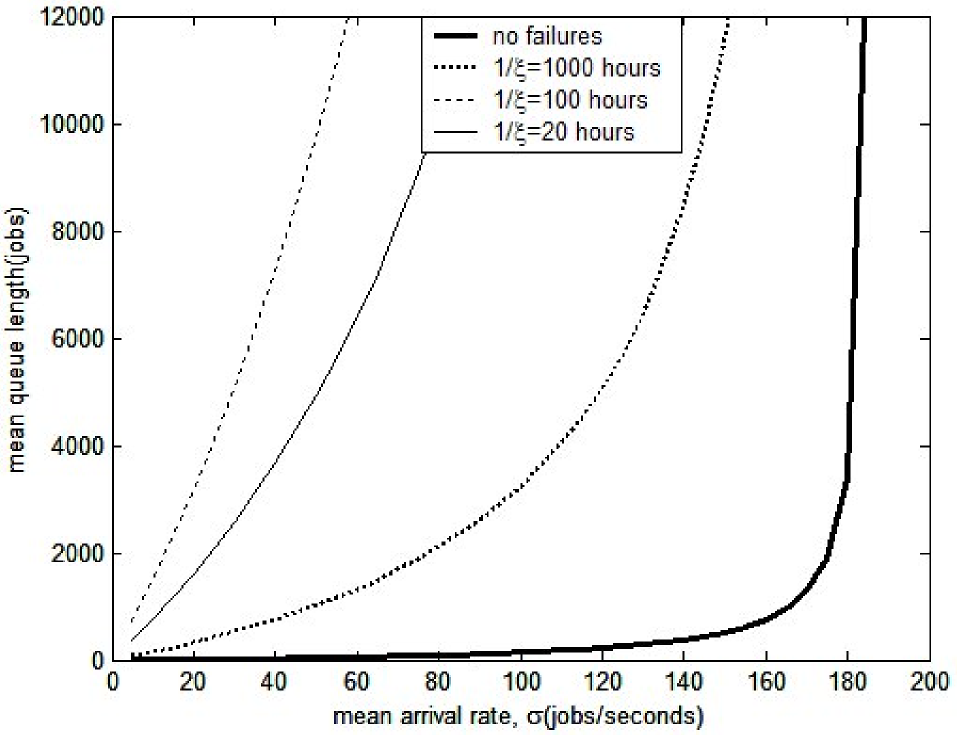


Figure 6.9: $q_{1,1}$ as a function of φ for various δ values



(a) Effects of ξ and σ for $1/\varphi$=70 seconds



(b) Effects of ξ and σ for $1/\varphi$=10 seconds

Figure 6.10: Effects of ξ and σ for $1/\varphi$ values

6.4 Conclusion

This chapter is concerned with modelling networks for performability evaluation to evaluate the effects of protocols proposed in Sections 5.2 and 5.3.

Security is further improved by frequently renewing the key distributed by the authentication server (Section 5.3) and during this time, restricting access to the system. The proposed approach has implications in the performance of the underlying network. Although it is expected that increased security will result in degradation in system performance, it is essential to ensure that any degradation in system performance is at acceptable levels. For this purpose, a mathematical model has been developed in Section 6.2 to evaluate the performance of the system considered and the mean queue lengths are calculated using various values for important system parameters. Performance degradation of the proposed system has been evaluated for various key distribution times while user access to the servers are shut down at certain intervals. Results enable designers to carefully choose parameters for the best MQL results for acceptable levels of security.

Section 6.3 is concerned with a modelling approach for performability evaluation of Kerberos servers which dynamically renew keys under pseudo-secure conditions. In order to evaluate the cost in terms of the degradation of system performance, an analytical method is used. Numerical results have been obtained and presented for various performability measures for different key renewal and interruption period values. Unlike the previous studies, the server failures are considered as well. Therefore the approach presented in this study provides more realistic performability measures. Results show that the server failures, as well as key renewal times and time between interruptions can significantly affect the performance of a Kerberos server especially if high arrival rates are expected. The model developed is highly flexible and it can be used for systems with various failure, repair, and renewal times and times between interruptions. The method can be extended for multiple Kerberos servers and for systems with backup servers especially for the KDC.

Chapter 7

Modelling Attacker with Increased Powers and Deciding Security Properties of Proposed Protocols by Induction and Deduction

7.1 Introduction

The work presented in this chapter is on constructing attacker with increased powers by inductions and deductions depending on proposed protocol specifications presented in Sections 3.2, 4.2 and 5.2.

The behaviour of agents while executing protocol steps in the presence of the attacker that constructed with inductive and deductive approaches, is the main concern of this chapter.

7.2 Modelling Attackers

Attackers and their potential in communication networks have been pictured in Section 2.3.1. The attack analysis is discussed prior to presenting a new model of an attacker. The new model developed is subsequently used to test proposed authentication protocols in the later sections of the chapter.

Formal methods for verifying security properties of cryptographic systems, designed for the purposes of assuring that systems satisfy their respective security requirements, tend towards analysing attacker potentials [Lowe, 1996], [Paulson, 1998], [Roscoe *et al.*, 2009]. In these publications, authors support the idea of knowing the attacker. The approaches of deductive, inductive or both, are based on making inferences about the attacker with a view to provide security offerings that can resist the activities of attackers. In the field of attack analysis, a pioneering model of an attacker is known as the Dolev Yao model [Dolev and Yao, 1981]. In this model an attacker is designed to be able to compose, intercept, and replay messages in all directions.

Induction is usually described as moving from the specific to the general, while deduction begins with the general and ends with the specific. Arguments based on laws rules and accepted principles are generally used for deductive arguments. Observations tend to be used for inductive arguments. In other words, in a deductive argument, the premises are intended to provide support for the conclusion that is so strong that, if the premises are true, it would be impossible for the conclusion to be false. An argument is a connected series of statements or propositions, some of which are intended to provide support, justification, or evidence for the truth of another statement or proposition. Arguments consist of one or more premises and a conclusion. Whereas, an inductive argument is an argument in which it is thought that the premises provide reasons supporting the probable truth of the conclusion. In an inductive argument, the premises are intended only to be so strong that, if they are true, then it is unlikely that the conclusion is false. Induction rule is defined such that every derivable judgment is the consequence of some rule, whose premises are derivable [Paulson, 1998] and [Ryan *et al.*, 2000].

In general verification tools [Hoare, 1985] that use the FDR model checker, with state space verification systems [Dolev and Yao, 1981], [Milner, 1999], attacker potentials are considered with respect to possible deductions. It is widely acknowledged that model checking offers more detailed analysis and discovers attacks faster [Roscoe, 2005] and [Roscoe *et al.*, 2009].

Following these discussions, a paradigm of verifying protocols by formal methods of induction was presented in [Paulson, 1998] using the proof tool *Isabelle/HOL*. This method includes theories of message analysis and of describing standard protocol features. Operations on sets of messages are inductively defined as the least set closed under specific extensions of a message set, H: where H contains an agent's initial knowledge and the history of all messages sent in a trace. Protocols are defined inductively as a set of traces. A trace is a list of communication events such as interleaved protocol runs, that is discussed in Section 2.3.1. The protocol description incorporates those of attacks and accidental losses. The attacker is modelled to know some private keys and can forge messages using components decrypted from previous traffic. The inductive operators used are parts, *analz*, and *synth*, which are defined on possibly infinite sets of messages. Thus, inductive verification copes with space explosion problem. Parts merely returns all the components of a set of messages, *analz* models the decryption of past traffic using available keys, and *synth* models the forging of messages. Several simplifying assumptions were used for protocols' description such as *uncrackable* cryptosystems, unambiguous data types, and rules bounding the enemy's capabilities. The model of the fraudulent messages that a spy, enemy, could derive from the set of messages H is presented by *synth(analz H)*; where *synth* and *analz* are operators. The first, *synth*, models messages a spy could build up from H, and *analz* models messages obtained by repeatedly adding compound messages and decrypting messages whose keys are in *analz H*.

With respect to the above descriptions, and discussions for the attacker model and intruder's inference rule on Section 2.3.1, the attacker modelled using possible deductions and inductions are presented as follow:

$$\begin{aligned}\text{Attacker(X)} &= \text{learn?m:M} \longrightarrow \text{Attacker(infer(X)} \cup \text{m)}\\ &\qquad \Box\text{say?m} : \text{X} \cap \text{M} \longrightarrow \text{Attacker(X)}\end{aligned}$$

$m \in M$ is a *facts* of set of messages. The function *infer(X)* generates the facts that are deducible from X under cryptographical possibilities. The above definition shows that the attacker has a choice of either to *learn* or to *say* messages. When the attacker learns a message of known type *M*, then it behaves as an attacker under deductive rules. Alternatively, the attacker can say a message through a fake channel, and continues to be present as an attacker at all times.

The paradigm of an inductive definition and operator induced by an inductive definition are based on [Cousot and Cousot, 1992] mathematical framework. The Attacker(Y) is described as:

$$\begin{aligned}\text{Attacker(Y)} &= \text{learn?m:M} \longrightarrow \text{Attacker(induce(Y)} \cup \text{m)}\\ &\qquad \Box\ \text{say?m} : \text{Y} \cap \text{M} \longrightarrow \text{Attacker(Y)}\end{aligned}$$

The function *induce(Y)* generates all facts that an intruder can *induce* from the message sets when they appear as partial orders or joins, which would require several steps of deductions to be calculated when using the function *infer*. Recall that *infer(X)* is defined in terms of possible deductions.

The above attacker definitions are built into the CSP models of proposed protocols given in Sections 3.2, 4.2, 5.2 and 5.3, and subsequently tested using the FDR. The FDR model checker tests proposed protocols by comparing *protocol description* and *specification*, where *protocol description* fails to be a refinement of the *specification* corresponds to attack against the protocol investigated. The next section looks at intruder in relation to specifications of proposed protocols.

7.3 Attacker with Increased Powers on Proposed Protocols

This section presents attackers that involve the authentication protocols in Sections 3.2, 4.2, 5.2 and 5.3. In terms of authentication, this section is concerned with increasing power of an intruder to break an encryption with the use of inductive as well as deductive techniques. Results of times taking to break the encryption for the designed protocols and intruder's power are presented and discussed.

7.3.1 Attacker for Proposed Framework

In this section, the analysis and verification are done through the use of FDR and CSP respectively. The "#Protocol description" section of the proposed protocol given in Section 3.3 is not modified, however, the "#Intruder Information" section is modified with knowledge. The following script of the protocol shows these changes.

```
#Protocol description
0. -> A : B
[B != A]
1. -> A : S
2. A -> S : B
3. S -> A : {{ts, B, kab}{SKey(A)}}{passwd(A,B)}
4. S -> A : {{ts, A, kab}{SKey(B)}}{passwd(A,B)} % enc
5. A -> B : enc % {{ts, A, kab}{SKey(B)}}{passwd(A,B)}
6. A -> B : {A, ta, na}{kab}
7. B -> A : {ta, na, nb}{kab}
8. A -> B : {nb, ta}{kab}
```

```
#Intruder Information
Intruder = Mallory
IntruderKnowledge = {Alice, Bob, Mallory, Sam, Nm, Km, PK, SK(Mallory), \
SKey(Mallory),passwd(Mallory,Alice), passwd(Mallory,Bob), \
passwd(Alice,Mallory),passwd(Bob,Mallory),passwd(Mallory,Mallory)}
Guessable = Password
Guessable = SessionKey
```

`IntruderKnowledge = {..., Km,.., \ }` `Guessable = SessionKey` are the entities added to intruder's knowledge.

The CSP models of the legitimate principals and the communication channels with message transformations were generated using CASPER. The intruder model with deductions, the feasibility of the protocol and the achievements of the *intruder* with deductive knowledge are demonstrated.

Intruder is verified with the following CSP code and through the analysis with the FDR checker, which is shown in the Figure 7.1 proves that intruder is not strong enough to decrypt the designed protocol's specifications.

```
IK0_init = union({Alice, Bob, Mallory, Sam, Nm, Km, SK(Mallory), SKey(Mallory),
             passwd(Mallory, Alice), passwd(Mallory, Bob),
             passwd(Alice, Mallory), passwd(Bob, Mallory),
             passwd(Mallory, Mallory), Garbage}, TimeStamp)
Guessable0 = Password
Guessable = diff(Guessable0, IK1)
```

Intruder has an initial knowledge of *Guessable* values and entities. Referring to the following script, the intruder closes up its initial knowledge under deductions and calculates the *facts* that can not be learnt.

```
components_(Sq.ms_) =
    if member(Sq.ms_, Fact_1) then {Sq.ms_} else set(ms_)
  components_(m_) = {m_}
Seeable_ =
    Union({unknown_(components_(m_)) | (_,m_,_,_) <- SYSTEM_M::INT_MSG_INFO})
Close_(IK_, ded_, fact_) =
    let IK1_ =
          union(IK_, {f_ | (f_,l_,fs_) <- ded_, fs_ <= IK_})
        ded1_ =
          {(f_,l_,fs_) | (f_,l_,fs_) <- ded_, fs_ <= fact_}
        fact1_ = Union({IK_, {f_ | (f_,l_,fs_) <- ded_},
              Seeable_, Guessable0})
```

Components of the intruder are grouped as *currently unknown **fact** f_:* and *known **fact** f_:*.

```
IGNORANT(f_,ms_,fss_,ds_) =
    hear?m_:ms_ -> KNOWS(f_,ms_,fss_,ds_)
    []
    ([] (l_, fs_) : fss_, not(member(f_,fs_)) @
        infer.(f_,l_,fs_) -> KNOWS(f_,ms_,fss_,ds_))
    []
    (member(f_,Guessable) & guess.f_ -> KNOWS''(f_,ms_,fss_,ds_))
    []
    guess?g_:diff(Guessable,{f_}) -> IGNORANT'(f_,ms_,fss_,ds_)

f_ms_fss_ds_s =
    let rid_ = relational_image({(f_,(l_,fs_)) | (f_,l_,fs_) <- Deductions})
        msf_ = relational_image({(f_, m_) | m_ <- MSG_BODY, f_ <- unSq_(m_)})
        xsf_ = relational_image({(f_, x_) | x_@@(_,l_,fs_) <- Deductions,
                                            f_ <- fs_})
    within {(f_, msf_(f_), rid_(f_), xsf_(f_)) | f_ <- KnowableFact}
```

leak is used to signal that a possible secret has been learnt, where *hear* and *say* are used to represent *hearing* or *saying* a message respectively. *infer(f,fs)* represent deducing **fact** *f* from the set of **facts** *fs*. *ms_* is the set of messages that contain *f_* at the top level, where *fss_* is the set of sets of **facts** from which *f_* can be deduced. *ds_* is the set of *deductions* that use *f_*.

```
KNOWS(f_,ms_,fss_,ds_) =
    hear?m_:ms_ -> KNOWS(f_,ms_,fss_,ds_)
    []
    say?m_:ms_ -> KNOWS(f_,ms_,fss_,ds_)
    []
    ([] ded@@(f1_,l_,fs_) : ds_, f1_!=f_ @ infer.ded -> KNOWS(f_,ms_,fss_,ds_))
    []
    member(f_,ALL_SECRETS_DI) & leak.f_ -> KNOWS(f_,ms_,fss_,ds_)
    []
    ([] (l_,fs_) : fss_ @ infer.(f_,l_,fs_) -> KNOWS(f_,ms_,fss_,ds_))
    []
    guess?g_ -> KNOWS'(f_,ms_,fss_,ds_)
```

Observing the previous FDR results, the authentication specifications and the entities of the Intruder's *Initial Knowledge* have been changed with the deductive inference rule. The "#Protocol description" section is not modified.

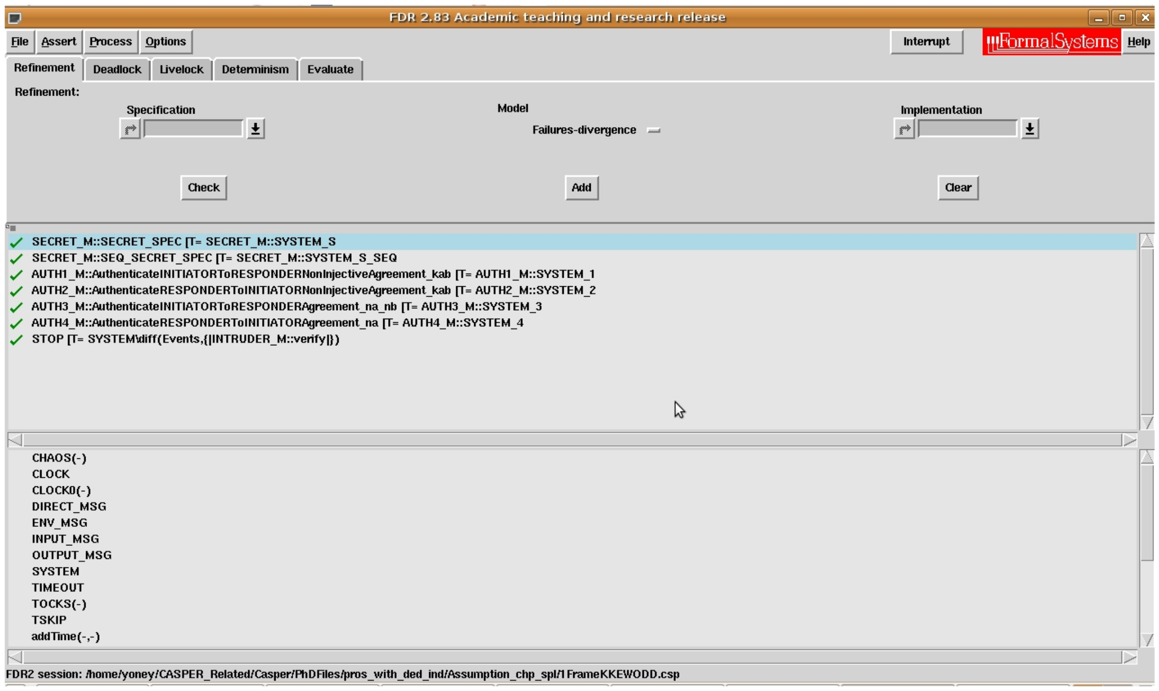
```
#Specification

NonInjectiveAgreement(A, B, [kab])
NonInjectiveAgreement(B, A, [kab])
Agreement(A, B, [na,nb])
Secret(A, passwd(A,B), [B])
Secret(A, kab, [B])
Agreement(B, A, [na])

#Intruder Information

Intruder = Mallory
IntruderKnowledge = {Alice, Bob, Mallory, Sam, Nm, Kab, PK, SK(Mallory), \
SKey(Mallory),passwd(Mallory,Alice), passwd(Mallory,Bob), \
passwd(Alice,Mallory),passwd(Bob,Mallory),passwd(Mallory,Mallory)}
Guessable = Password
Guessable = SessionKey
```

`IntruderKnowledge = {Alice, Bob, Mallory, Sam, Nm, Kab, PK, SK(Mallory),\` holds the new entity. Instead of known varible *SessionKey*, `Km`, another SessionKey, `Kab` is defined, with the knowledge of `Guessable = SessionKey`.

Intruder's capabilities are analysed with the FDR checker, and the results are shown in Figure 7.2 where the verification for *intruder* with deductive knowledge is provided for each authentication specification. The outline of the FDR results that follows the intruder named *Mallory*, while the legitimate agents INITIATOR and RESPONDER are *Alice* and *Bob* respectively.

During the analysis where assertions failed two levels of counter examples were checked from the process tree of the FDR **debugger**; thus, the first is namely $SYSTEM\_1$ and the second one is *SYSTEM*. As it is expressed in Sections 4.3 and 5.3, the failed assertions correspond to instances of attacks. The analysis performed is presented under deductions followed by deductions allocated with inductions.

When intruder ability is based on only deduction, the authentication specifications `Agreement(A,B,[na,nb])`, `NonInjectiveAgreement(A,B,[kab])`, and `NonInjectiveAgreement(B,A,[kab])` failed when deduction process was allocated with the induction process. The observed results from the FDR **debugger** are as follow in Figures 7.3, 7.4, and 7.5.
In a similar way, intruder definition based on deduction with possible induction, the secrecy specifications `Secret(A, passwd(A,B), [B])` and `Secret(A, kab, [B])` showed failures in Figures 7.6 and 7.7.

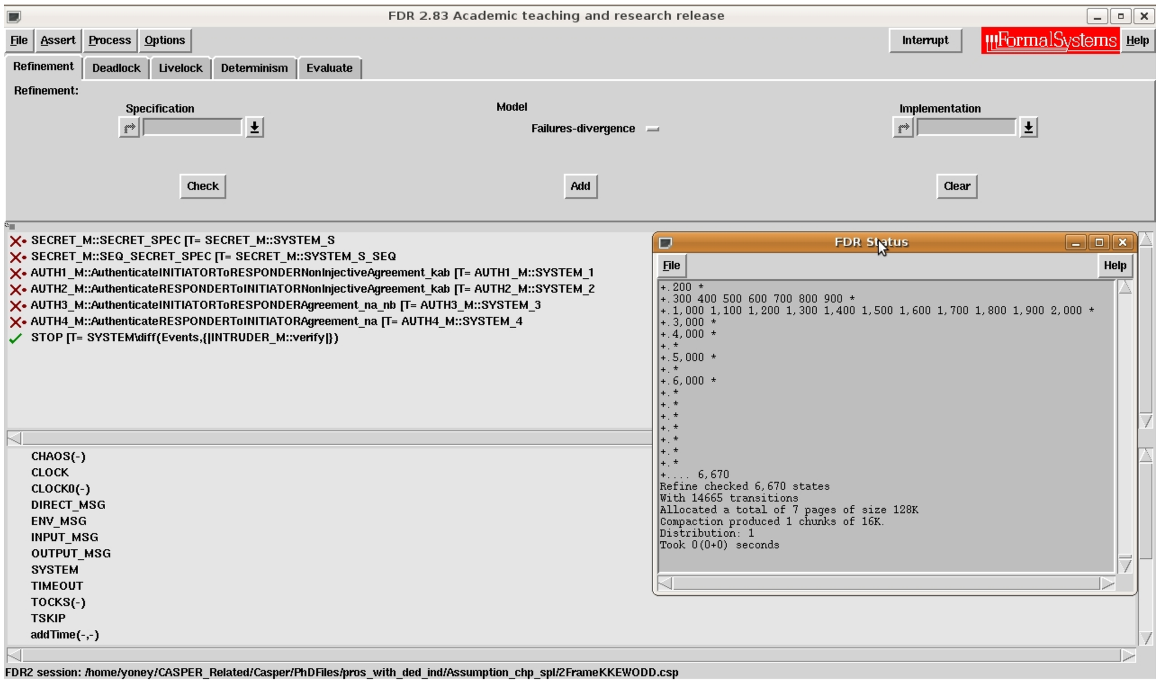


```
✓ SECRET_M::SECRET_SPEC [T= SECRET_M::SYSTEM_S
✓ SECRET_M::SEQ_SECRET_SPEC [T= SECRET_M::SYSTEM_S_SEQ
✓ AUTH1_M::AuthenticateINITIATORToRESPONDERNonInjectiveAgreement_kab [T= AUTH1_M::SYSTEM_1
✓ AUTH2_M::AuthenticateRESPONDERToINITIATORNonInjectiveAgreement_kab [T= AUTH2_M::SYSTEM_2
✓ AUTH3_M::AuthenticateINITIATORToRESPONDERAgreement_na_nb [T= AUTH3_M::SYSTEM_3
✓ AUTH4_M::AuthenticateRESPONDERToINITIATORAgreement_na [T= AUTH4_M::SYSTEM_4
✓ STOP [T= SYSTEM\diff(Events,{|INTRUDER_M::verify|})
```

```
CHAOS(-)
CLOCK
CLOCK0(-)
DIRECT_MSG
ENV_MSG
INPUT_MSG
OUTPUT_MSG
SYSTEM
TIMEOUT
TOCKS(-)
TSKIP
addTime(-,-)
```

FDR2 session: /home/yoney/CASPER_Related/Casper/PhDFiles/pros_with_ded_ind/Assumption_chp_spl/1FrameKKEWODD.csp

Figure 7.1: Achivement of intruder on the proposed framework



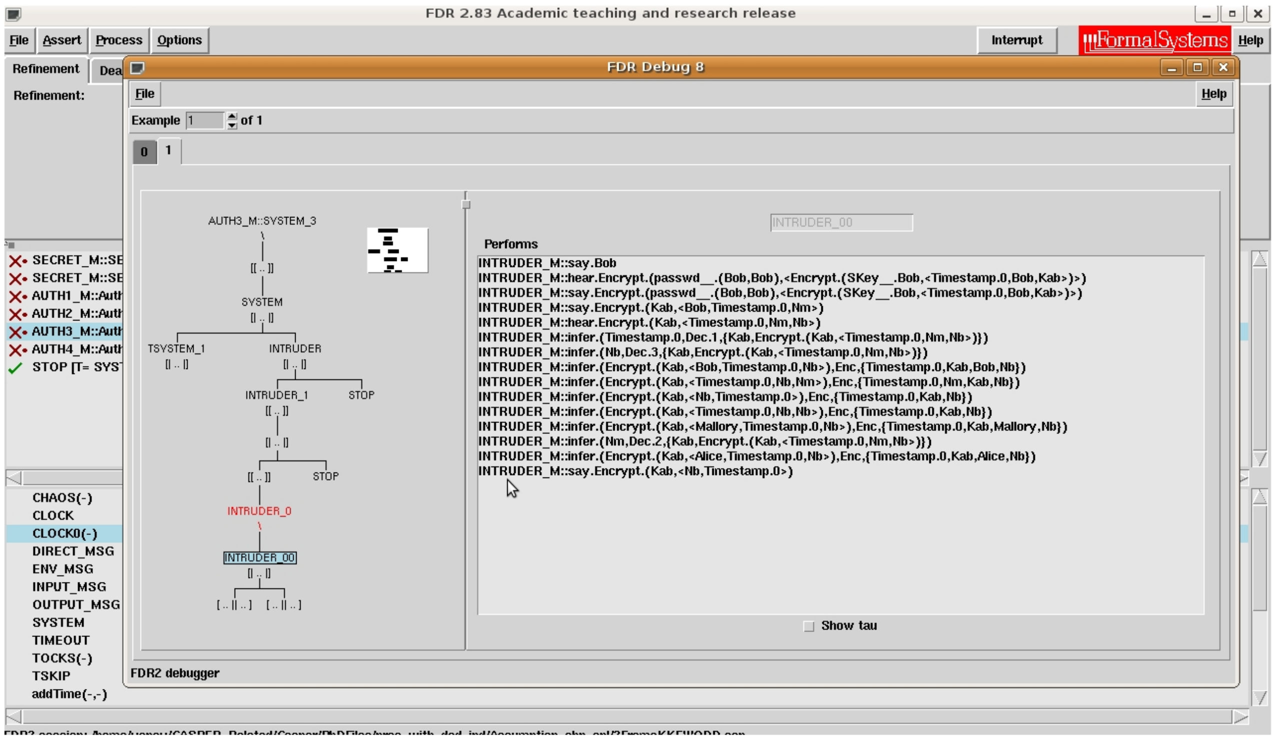
```
✗ SECRET_M::SECRET_SPEC [T= SECRET_M::SYSTEM_S
✗ SECRET_M::SEQ_SECRET_SPEC [T= SECRET_M::SYSTEM_S_SEQ
✗ AUTH1_M::AuthenticateINITIATORToRESPONDERNonInjectiveAgreement_kab [T= AUTH1_M::SYSTEM_1
✗ AUTH2_M::AuthenticateRESPONDERToINITIATORNonInjectiveAgreement_kab [T= AUTH2_M::SYSTEM_2
✗ AUTH3_M::AuthenticateINITIATORToRESPONDERAgreement_na_nb [T= AUTH3_M::SYSTEM_3
✗ AUTH4_M::AuthenticateRESPONDERToINITIATORAgreement_na [T= AUTH4_M::SYSTEM_4
✓ STOP [T= SYSTEM\diff(Events,{|INTRUDER_M::verify|})
```

```
+.200 *
+.300 400 500 600 700 800 900 *
+.1,000 1,100 1,200 1,300 1,400 1,500 1,600 1,700 1,800 1,900 2,000 *
+.3,000 *
+.4,000 *
+.*
+.5,000 *
+.*
+.6,000 *
+.*
+.*
+.*
+.*
+.*
+.*
+.*
+.... 6,670
Refine checked 6,670 states
With 14665 transitions
Allocated a total of 7 pages of size 128K
Compaction produced 1 chunks of 16K.
Distribution: 1
Took 0(0+0) seconds
```

```
CHAOS(-)
CLOCK
CLOCK0(-)
DIRECT_MSG
ENV_MSG
INPUT_MSG
OUTPUT_MSG
SYSTEM
TIMEOUT
TOCKS(-)
TSKIP
addTime(-,-)
```

FDR2 session: /home/yoney/CASPER_Related/Casper/PhDFiles/pros_with_ded_ind/Assumption_chp_spl/2FrameKKEWODD.csp

Figure 7.2: Power of intruder with deductions for the proposed framework



```
Performs
INTRUDER_M::say.Bob
INTRUDER_M::hear.Encrypt.(passwd__.(Bob,Bob),<Encrypt.(SKey__.Bob,<Timestamp.0,Bob,Kab>)>)
INTRUDER_M::say.Encrypt.(passwd__.(Bob,Bob),<Encrypt.(SKey__.Bob,<Timestamp.0,Bob,Kab>)>)
INTRUDER_M::say.Encrypt.(Kab,<Bob,Timestamp.0,Nm>)
INTRUDER_M::hear.Encrypt.(Kab,<Timestamp.0,Nm,Nb>)
INTRUDER_M::infer.(Timestamp.0,Dec.1,{Kab,Encrypt.(Kab,<Timestamp.0,Nm,Nb>)})
INTRUDER_M::infer.(Nb,Dec.3,{Kab,Encrypt.(Kab,<Timestamp.0,Nm,Nb>)})
INTRUDER_M::infer.(Encrypt.(Kab,<Bob,Timestamp.0,Nb>),Enc,{Timestamp.0,Kab,Bob,Nb})
INTRUDER_M::infer.(Encrypt.(Kab,<Timestamp.0,Nb,Nm>),Enc,{Timestamp.0,Nm,Kab,Nb})
INTRUDER_M::infer.(Encrypt.(Kab,<Nb,Timestamp.0>),Enc,{Timestamp.0,Kab,Nb})
INTRUDER_M::infer.(Encrypt.(Kab,<Timestamp.0,Nb,Nb>),Enc,{Timestamp.0,Kab,Nb})
INTRUDER_M::infer.(Encrypt.(Kab,<Mallory,Timestamp.0,Nb>),Enc,{Timestamp.0,Kab,Mallory,Nb})
INTRUDER_M::infer.(Nm,Dec.2,{Kab,Encrypt.(Kab,<Timestamp.0,Nm,Nb>)})
INTRUDER_M::infer.(Encrypt.(Kab,<Alice,Timestamp.0,Nb>),Enc,{Timestamp.0,Kab,Alice,Nb})
INTRUDER_M::say.Encrypt.(Kab,<Nb,Timestamp.0>)
```

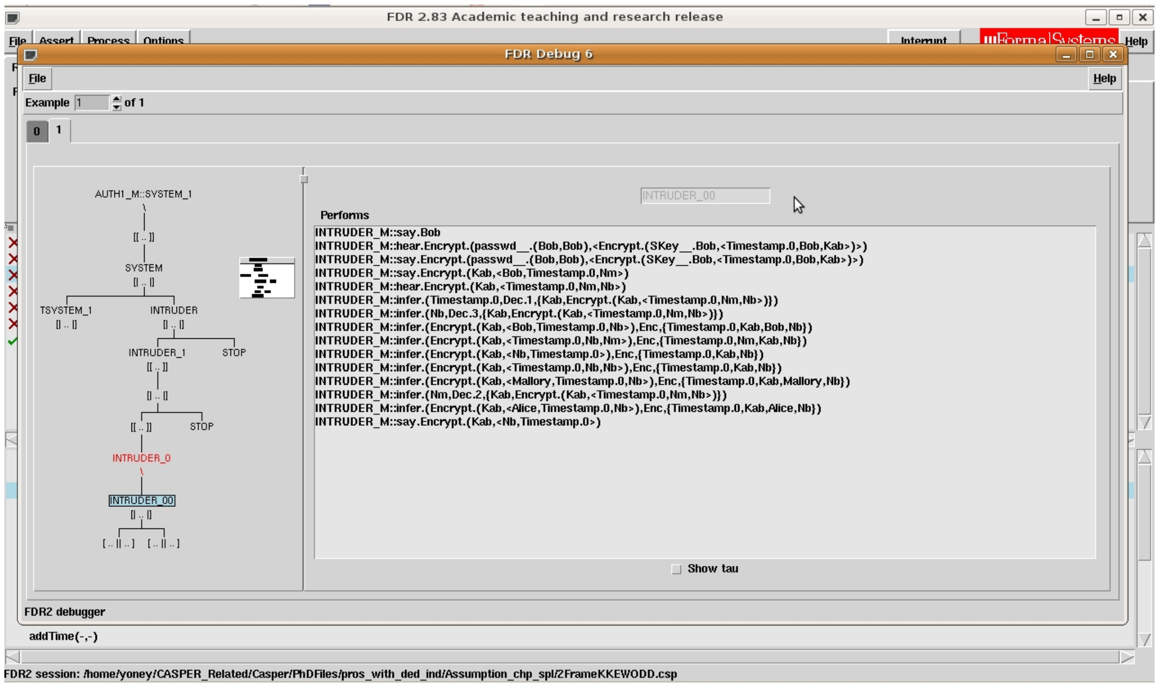
```
SECRET_M::SE
SECRET_M::SE
AUTH1_M::Auth
AUTH2_M::Auth
AUTH3_M::Auth
AUTH4_M::Auth
STOP [T= SYS
CHAOS(-)
CLOCK
CLOCK0(-)
DIRECT_MSG
ENV_MSG
INPUT_MSG
OUTPUT_MSG
SYSTEM
TIMEOUT
TOCKS(-)
TSKIP
addTime(-,-)
```

Figure 7.3: Power of intruder on Agreement(A,B,[na,nb]) specification

Thus, addition of induction reveals a fresh attack against the *INITIATOR* and *RESPONDER*, which was not possible with intruder ability defined under possible deductions only. The attack is discussed in Section 3.3. Secondly, it is observed that the addition of induction has significantly reduced the number of steps registered for the respective steps.

7.3.2 Attacker for the Proposed Protocol with Delayed Decryption Property

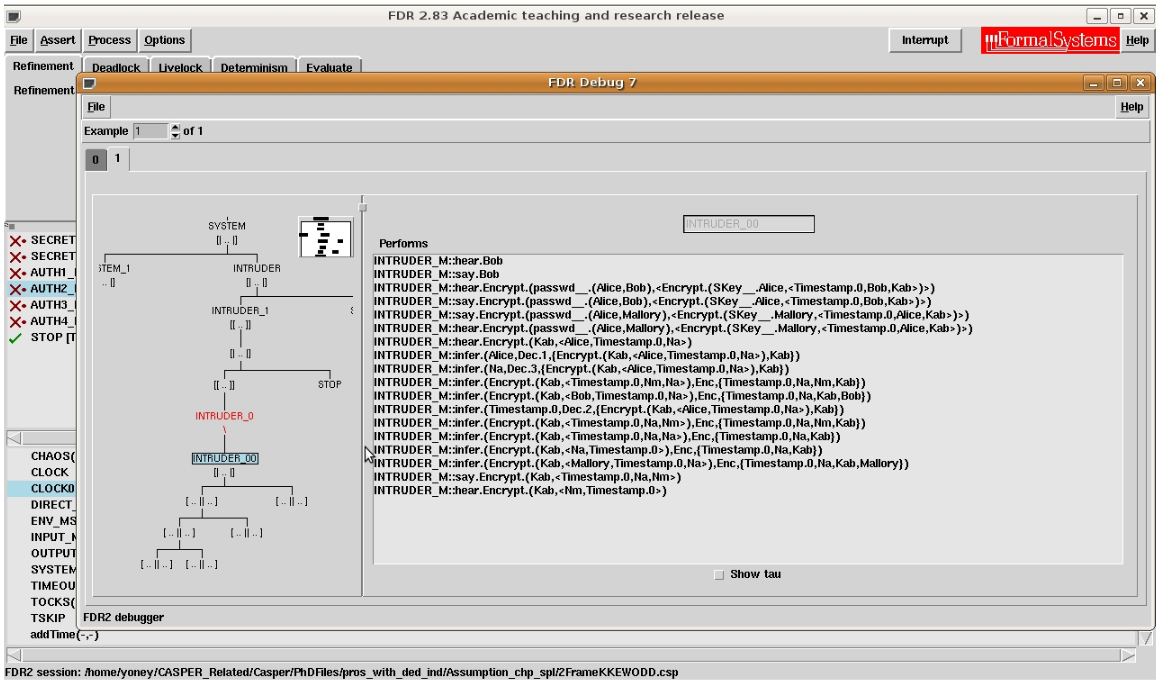
In this section, the analysis and verification are done through the use of FDR and CSP respectively. The "#Protocol description" section of the proposed protocol given in Section 4.2 is not modified, however the "#Intruder Information" section is modified with knowledge. Following script of the protocol shows the changes on the intruder.


FDR 2.83 Academic teaching and research release
FDR Debug 6
File
Help
Example 1 of 1
AUTH1_M::SYSTEM_1
SYSTEM
TSYSTEM_1
INTRUDER
INTRUDER_1
STOP
INTRUDER_0
INTRUDER_00
Show tau
FDR2 debugger
addTime(-,-)
FDR2 session: /home/yoney/CASPER_Related/Casper/PhDFiles/pros_with_ded_ind/Assumption_chp_spl/2FrameKKEWODD.csp

Performs

```
INTRUDER_M::say.Bob
INTRUDER_M::hear.Encrypt.(passwd__.(Bob,Bob),<Encrypt.(SKey__.Bob,<Timestamp.0,Bob,Kab>)>)
INTRUDER_M::say.Encrypt.(passwd__.(Bob,Bob),<Encrypt.(SKey__.Bob,<Timestamp.0,Bob,Kab>)>)
INTRUDER_M::say.Encrypt.(Kab,<Bob,Timestamp.0,Nm>)
INTRUDER_M::hear.Encrypt.(Kab,<Timestamp.0,Nm,Nb>)
INTRUDER_M::infer.(Timestamp.0,Dec.1,{Kab,Encrypt.(Kab,<Timestamp.0,Nm,Nb>)})
INTRUDER_M::infer.(Nb,Dec.3,{Kab,Encrypt.(Kab,<Timestamp.0,Nm,Nb>)})
INTRUDER_M::infer.(Encrypt.(Kab,<Bob,Timestamp.0,Nb>),Enc,{Timestamp.0,Kab,Bob,Nb})
INTRUDER_M::infer.(Encrypt.(Kab,<Timestamp.0,Nb,Nm>),Enc,{Timestamp.0,Nm,Kab,Nb})
INTRUDER_M::infer.(Encrypt.(Kab,<Nb,Timestamp.0>),Enc,{Timestamp.0,Kab,Nb})
INTRUDER_M::infer.(Encrypt.(Kab,<Timestamp.0,Nb,Nb>),Enc,{Timestamp.0,Kab,Nb})
INTRUDER_M::infer.(Encrypt.(Kab,<Mallory,Timestamp.0,Nb>),Enc,{Timestamp.0,Kab,Mallory,Nb})
INTRUDER_M::infer.(Nm,Dec.2,{Kab,Encrypt.(Kab,<Timestamp.0,Nm,Nb>)})
INTRUDER_M::infer.(Encrypt.(Kab,<Alice,Timestamp.0,Nb>),Enc,{Timestamp.0,Kab,Alice,Nb})
INTRUDER_M::say.Encrypt.(Kab,<Nb,Timestamp.0>)
```

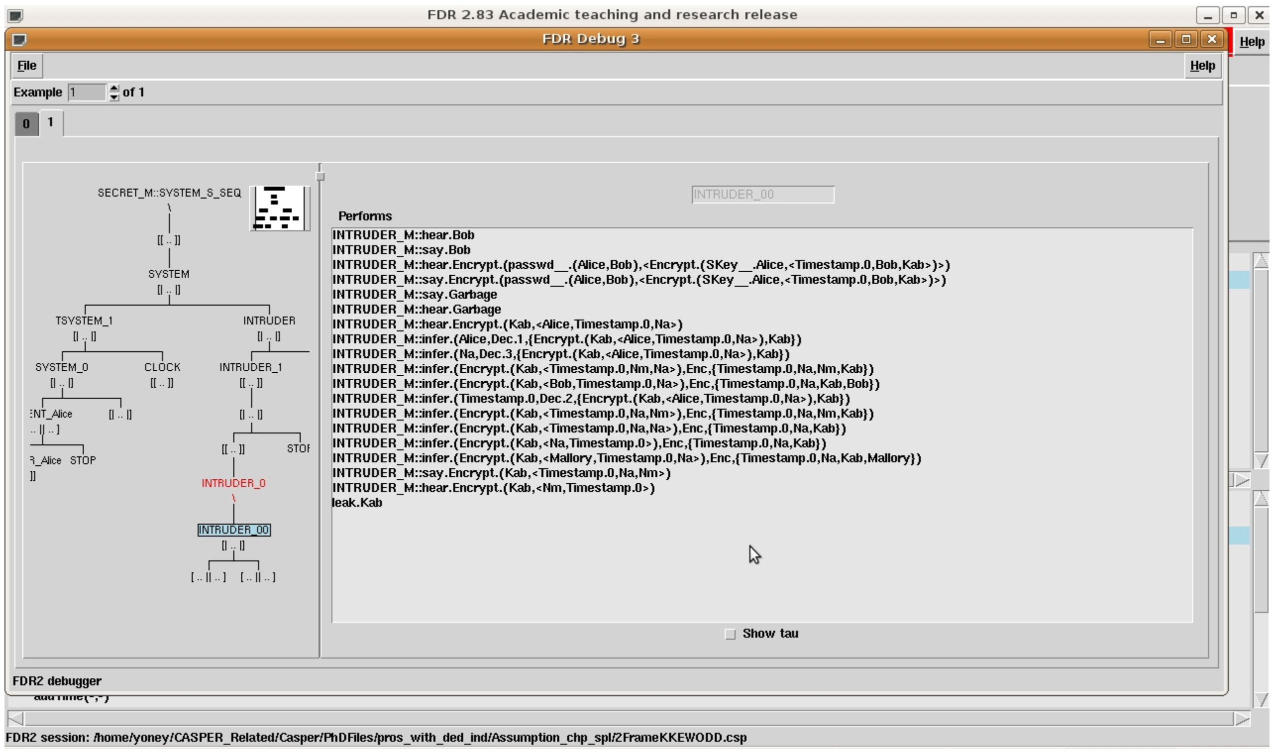
Figure 7.4: Power of intruder on NonInjectiveAgreement(A,B,[kab]) specification



Performs

```
INTRUDER_M::hear.Bob
INTRUDER_M::say.Bob
INTRUDER_M::hear.Encrypt.(passwd__.(Alice,Bob),<Encrypt.(SKey__.Alice,<Timestamp.0,Bob,Kab>)>)
INTRUDER_M::say.Encrypt.(passwd__.(Alice,Bob),<Encrypt.(SKey__.Alice,<Timestamp.0,Bob,Kab>)>)
INTRUDER_M::say.Encrypt.(passwd__.(Alice,Mallory),<Encrypt.(SKey__.Mallory,<Timestamp.0,Alice,Kab>)>)
INTRUDER_M::hear.Encrypt.(passwd__.(Alice,Mallory),<Encrypt.(SKey__.Mallory,<Timestamp.0,Alice,Kab>)>)
INTRUDER_M::hear.Encrypt.(Kab,<Alice,Timestamp.0,Na>)
INTRUDER_M::infer.(Alice,Dec.1,{Encrypt.(Kab,<Alice,Timestamp.0,Na>),Kab})
INTRUDER_M::infer.(Na,Dec.3,{Encrypt.(Kab,<Alice,Timestamp.0,Na>),Kab})
INTRUDER_M::infer.(Encrypt.(Kab,<Timestamp.0,Nm,Na>),Enc,{Timestamp.0,Na,Nm,Kab})
INTRUDER_M::infer.(Encrypt.(Kab,<Bob,Timestamp.0,Na>),Enc,{Timestamp.0,Na,Kab,Bob})
INTRUDER_M::infer.(Timestamp.0,Dec.2,{Encrypt.(Kab,<Alice,Timestamp.0,Na>),Kab})
INTRUDER_M::infer.(Encrypt.(Kab,<Timestamp.0,Na,Nm>),Enc,{Timestamp.0,Na,Nm,Kab})
INTRUDER_M::infer.(Encrypt.(Kab,<Timestamp.0,Na,Na>),Enc,{Timestamp.0,Na,Kab})
INTRUDER_M::infer.(Encrypt.(Kab,<Na,Timestamp.0>),Enc,{Timestamp.0,Na,Kab})
INTRUDER_M::infer.(Encrypt.(Kab,<Mallory,Timestamp.0,Na>),Enc,{Timestamp.0,Na,Kab,Mallory})
INTRUDER_M::say.Encrypt.(Kab,<Timestamp.0,Na,Nm>)
INTRUDER_M::hear.Encrypt.(Kab,<Nm,Timestamp.0>)
```

Figure 7.5: Power of intruder on NonInjectiveAgreement(B,A,[kab]) specification



```
Performs
INTRUDER_M::hear.Bob
INTRUDER_M::say.Bob
INTRUDER_M::hear.Encrypt.(passwd__.(Alice,Bob),<Encrypt.(SKey__.Alice,<Timestamp.0,Bob,Kab>)>)
INTRUDER_M::say.Encrypt.(passwd__.(Alice,Bob),<Encrypt.(SKey__.Alice,<Timestamp.0,Bob,Kab>)>)
INTRUDER_M::say.Garbage
INTRUDER_M::hear.Garbage
INTRUDER_M::hear.Encrypt.(Kab,<Alice,Timestamp.0,Na>)
INTRUDER_M::infer.(Alice,Dec.1,{Encrypt.(Kab,<Alice,Timestamp.0,Na>),Kab})
INTRUDER_M::infer.(Na,Dec.3,{Encrypt.(Kab,<Alice,Timestamp.0,Na>),Kab})
INTRUDER_M::infer.(Encrypt.(Kab,<Timestamp.0,Nm,Na>),Enc,{Timestamp.0,Na,Nm,Kab})
INTRUDER_M::infer.(Encrypt.(Kab,<Bob,Timestamp.0,Na>),Enc,{Timestamp.0,Na,Kab,Bob})
INTRUDER_M::infer.(Timestamp.0,Dec.2,{Encrypt.(Kab,<Alice,Timestamp.0,Na>),Kab})
INTRUDER_M::infer.(Encrypt.(Kab,<Timestamp.0,Na,Nm>),Enc,{Timestamp.0,Na,Nm,Kab})
INTRUDER_M::infer.(Encrypt.(Kab,<Timestamp.0,Na,Na>),Enc,{Timestamp.0,Na,Kab})
INTRUDER_M::infer.(Encrypt.(Kab,<Na,Timestamp.0>),Enc,{Timestamp.0,Na,Kab})
INTRUDER_M::infer.(Encrypt.(Kab,<Mallory,Timestamp.0,Na>),Enc,{Timestamp.0,Na,Kab,Mallory})
INTRUDER_M::say.Encrypt.(Kab,<Timestamp.0,Na,Nm>)
INTRUDER_M::hear.Encrypt.(Kab,<Nm,Timestamp.0>)
leak.Kab
```

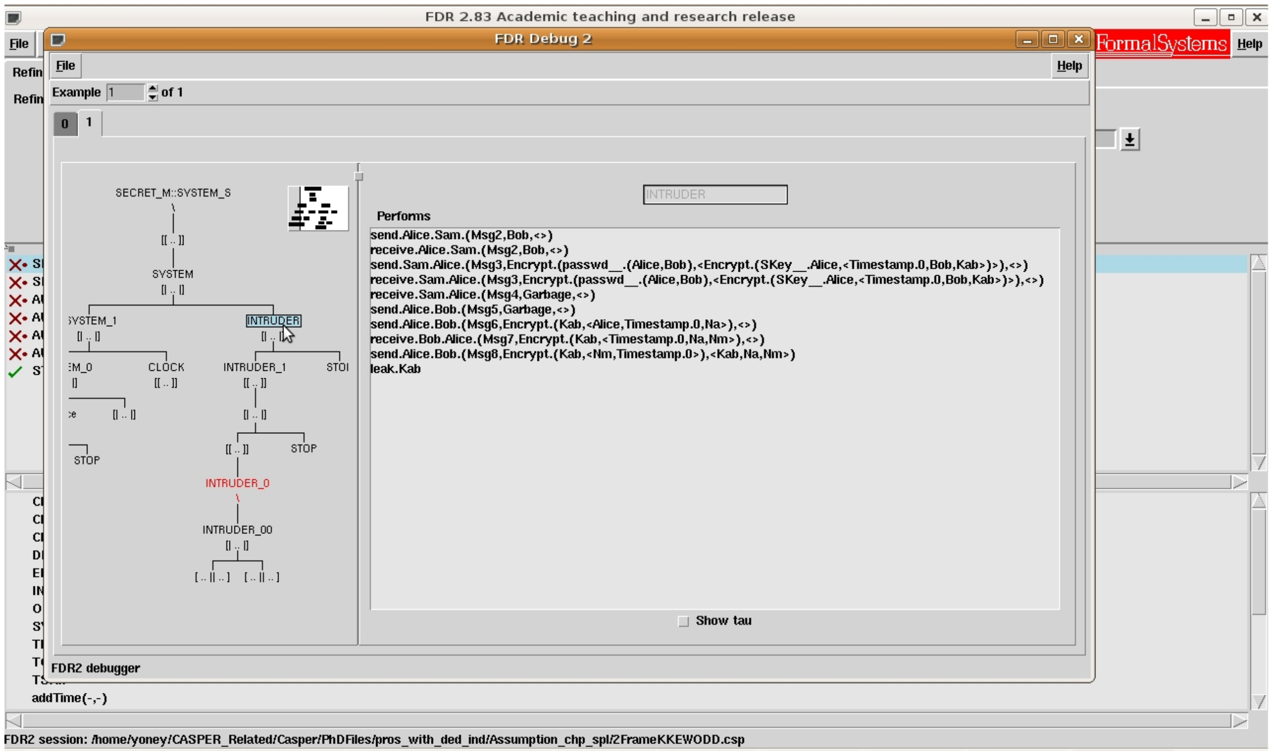
FDR2 session: /home/yoney/CASPER_Related/Casper/PhDFiles/pros_with_ded_ind/Assumption_chp_spl/2FrameKKEWODD.csp

Figure 7.6: Power of intruder on Secret(A,passwd(A,B),[B]) specification

```
#Intruder Information

Intruder = Mallory
IntruderKnowledge = {Alice, Bob, Mallory, Sam, Nm, Km,Kab PK, SPK, \
SK(Mallory), SKey(Mallory),passwd(Mallory,Alice), passwd(Mallory,Bob), \
passwd(Alice,Mallory), passwd(Bob,Mallory), passwd(Mallory,Mallory)}
Crackable = SessionKey
Crackable = ServerKey
Crackable = Password
```

`IntruderKnowledge = {..., Km,Kab.., \ } Crackable = SessionKey` are the entities added to the intruder's knowledge.

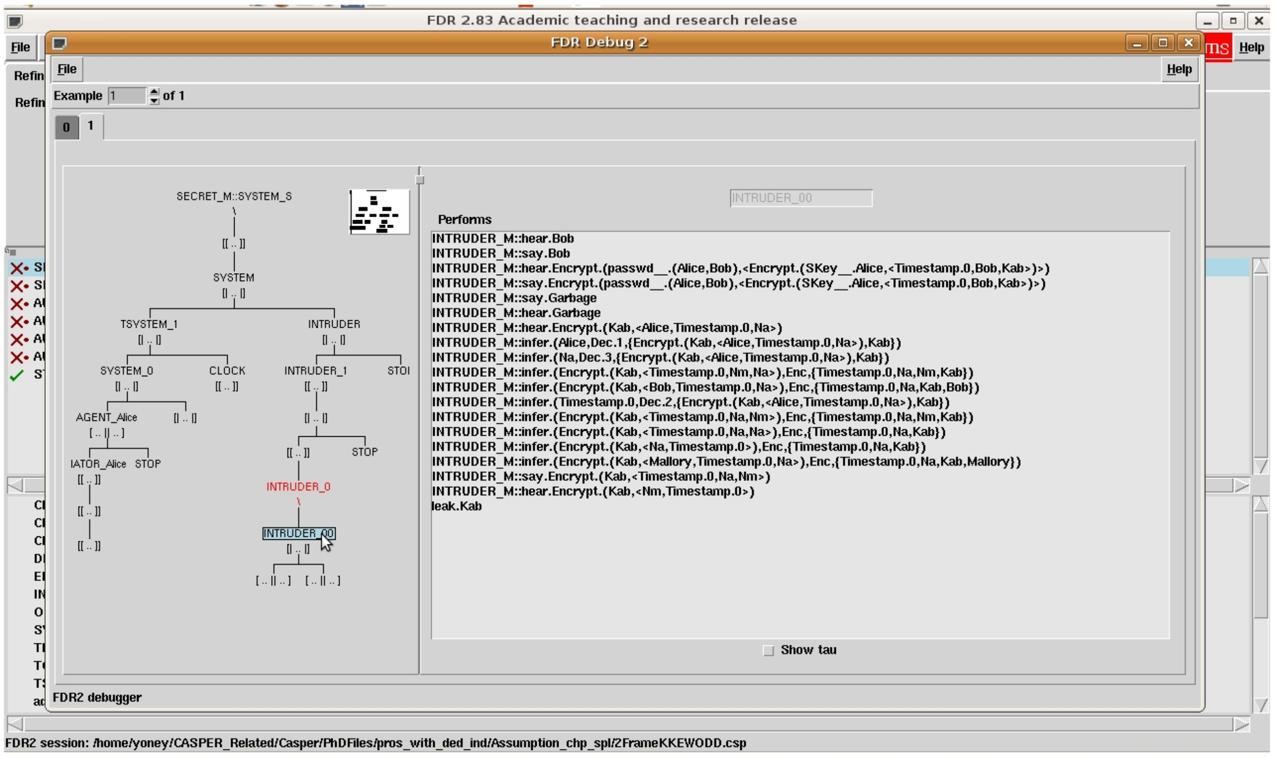
The CSP models of the legitimate principals and the communication channels with message transformations were generated using CASPER. The intruder model with deductions, the feasibility of the protocol and the achievements of *intruder* with deductive knowledge are demonstrated.



```
send.Alice.Sam.(Msg2,Bob,<>)
receive.Alice.Sam.(Msg2,Bob,<>)
send.Sam.Alice.(Msg3,Encrypt.(passwd__.(Alice,Bob),<Encrypt.(SKey__.Alice,<Timestamp.0,Bob,Kab>)>),<>)
receive.Sam.Alice.(Msg3,Encrypt.(passwd__.(Alice,Bob),<Encrypt.(SKey__.Alice,<Timestamp.0,Bob,Kab>)>),<>)
receive.Sam.Alice.(Msg4,Garbage,<>)
send.Alice.Bob.(Msg5,Garbage,<>)
send.Alice.Bob.(Msg6,Encrypt.(Kab,<Alice,Timestamp.0,Na>),<>)
receive.Bob.Alice.(Msg7,Encrypt.(Kab,<Timestamp.0,Na,Nm>),<>)
send.Alice.Bob.(Msg8,Encrypt.(Kab,<Nm,Timestamp.0>),<Kab,Na,Nm>)
leak.Kab
```

FDR2 session: /home/yoney/CASPER_Related/Casper/PhDFiles/pros_with_ded_ind/Assumption_chp_spl/2FrameKKEWODD.csp

(a)



```
INTRUDER_M::hear.Bob
INTRUDER_M::say.Bob
INTRUDER_M::hear.Encrypt.(passwd__.(Alice,Bob),<Encrypt.(SKey__.Alice,<Timestamp.0,Bob,Kab>)>)
INTRUDER_M::say.Encrypt.(passwd__.(Alice,Bob),<Encrypt.(SKey__.Alice,<Timestamp.0,Bob,Kab>)>)
INTRUDER_M::say.Garbage
INTRUDER_M::hear.Garbage
INTRUDER_M::hear.Encrypt.(Kab,<Alice,Timestamp.0,Na>)
INTRUDER_M::infer.(Alice,Dec.1,{Encrypt.(Kab,<Alice,Timestamp.0,Na>),Kab})
INTRUDER_M::infer.(Na,Dec.3,{Encrypt.(Kab,<Alice,Timestamp.0,Na>),Kab})
INTRUDER_M::infer.(Encrypt.(Kab,<Timestamp.0,Nm,Na>),Enc,{Timestamp.0,Na,Nm,Kab})
INTRUDER_M::infer.(Encrypt.(Kab,<Bob,Timestamp.0,Na>),Enc,{Timestamp.0,Na,Kab,Bob})
INTRUDER_M::infer.(Timestamp.0,Dec.2,{Encrypt.(Kab,<Alice,Timestamp.0,Na>),Kab})
INTRUDER_M::infer.(Encrypt.(Kab,<Timestamp.0,Na,Nm>),Enc,{Timestamp.0,Na,Nm,Kab})
INTRUDER_M::infer.(Encrypt.(Kab,<Timestamp.0,Na,Na>),Enc,{Timestamp.0,Na,Kab})
INTRUDER_M::infer.(Encrypt.(Kab,<Na,Timestamp.0>),Enc,{Timestamp.0,Na,Kab})
INTRUDER_M::infer.(Encrypt.(Kab,<Mallory,Timestamp.0,Na>),Enc,{Timestamp.0,Na,Kab,Mallory})
INTRUDER_M::say.Encrypt.(Kab,<Timestamp.0,Na,Nm>)
INTRUDER_M::hear.Encrypt.(Kab,<Nm,Timestamp.0>)
leak.Kab
```

FDR2 session: /home/yoney/CASPER_Related/Casper/PhDFiles/pros_with_ded_ind/Assumption_chp_spl/2FrameKKEWODD.csp

(b)

Figure 7.7: Power of intruder on Secret(A,kab,[B]) specification

```
#Specification

Secret(A, passwd(A,B), [B])
Agreement(A, B, [na])
Secret(A, kab, [B])
Secret(B, kab, [A])
```

Refering to the above protocol specification section, the specifications for the protocol are done for the legitimate principals, *INITIATOR* and *RESPONDER*. Two different types of specifications are defined for secrecy and authentication. The following CSP code is specification for single secret.

```
SECRET_SPEC_0(s_) =
    signal.Claim_Secret?A_!s_?Bs_ ->
      (if member(Mallory, Bs_) then SECRET_SPEC_0(s_)
       else SECRET_SPEC_1(s_))
    []
    leak.s_ -> SECRET_SPEC_0(s_)
    []
    crack?k_ -> SECRET_SPEC_0(s_)
  SECRET_SPEC_1(s_) =
    signal.Claim_Secret?A_!s_?Bs_ -> SECRET_SPEC_1(s_)
    []
    crack?k_ -> SECRET_SPEC_0(s_)
```

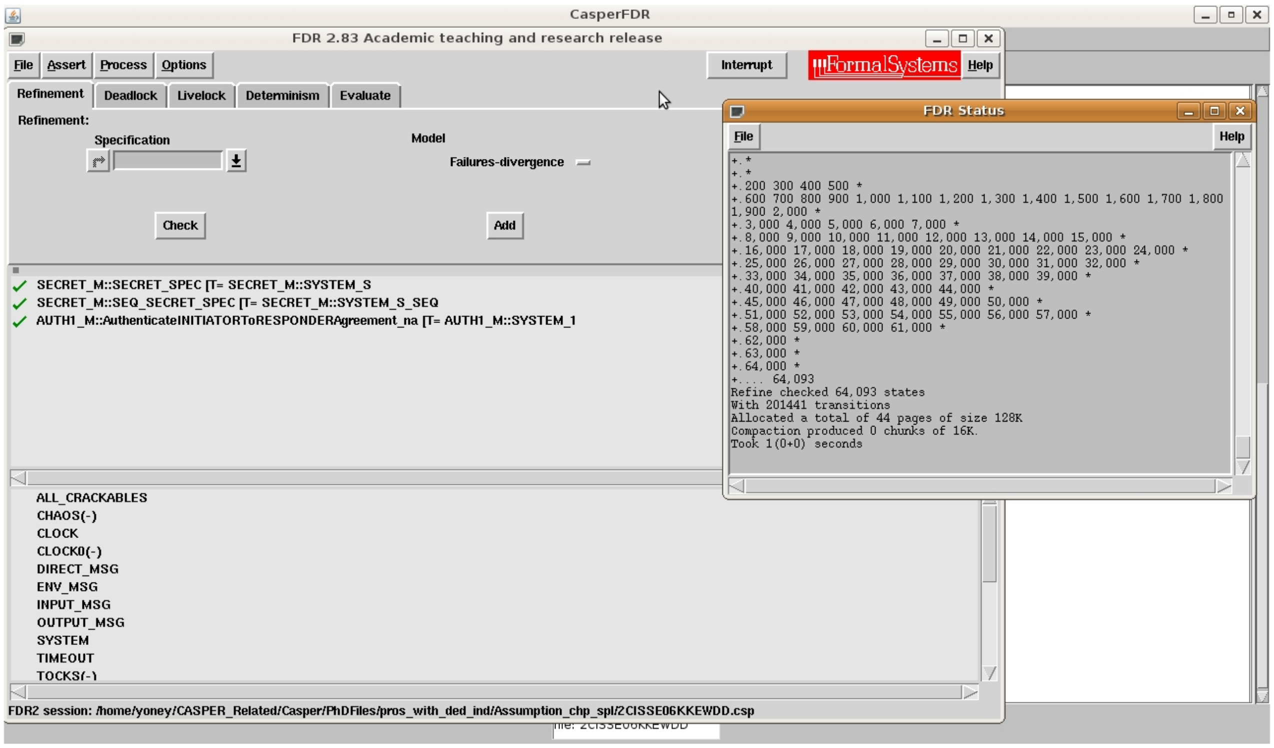
Addition to this specification, *sequential version*: *secs_* is secrets that intruder must not learn is defined with the following entities:

```
SEQ_SECRET_SPEC_0(secs_) =
    scs?s_!IntIn -> SEQ_SECRET_SPEC_0(secs_)
    []
    card(secs_)<2 & scs?s_!IntNotIn ->
      SEQ_SECRET_SPEC_0(union(secs_,{s_}))
    []
    card(secs_)==2 & scs?s_:secs_!IntNotIn ->
      SEQ_SECRET_SPEC_0(secs_)
    []
    leak?s_ : diff(ALL_SECRETS,secs_) -> SEQ_SECRET_SPEC_0(secs_)
    []
    crack?k_ -> SEQ_SECRET_SPEC_0(diff(secs_,{k_}))

isIntIn(S_) = if member(Mallory,S_) then IntIn else IntNotIn
```

Addition to the *secrecy* specifications, authentication specifications for all agents that are being authenticated is modelled as in CSP as follows:

```
AuthenticateINITIATORToRESPONDERAgreement_na =
    (AuthenticateINITIATORAliceToRESPONDERAgreement_na
    [| inter(AlphaAuthenticateINITIATORToRESPONDERAgreement_na_0(Alice),
             AlphaAuthenticateINITIATORToRESPONDERAgreement_na_0(Bob)) |]
    AuthenticateINITIATORBobToRESPONDERAgreement_na)
```

The intruder model is analysed with the FDR checker, which is shown in the Figure 7.8. The authentication agreement, `Agreement(A,B,[na])`, and the secret agreements, `Secret(A, passwd(A,B), [B])`, `Secret(A, kab, [B])`, and `Secret(B, kab, [A])` are not decrypted with the intruder definition being under the possible deduction; the implementation at this point was justified as a refinement of the specification. Since *no attack was found*, the FDR **debugger** produced *no traces*. This proves that intruder is not strong enough to decrypt the proposed protocol's specifications.



Refinement:
Specification
Model
Failures-divergence

✓ SECRET_M::SECRET_SPEC [T= SECRET_M::SYSTEM_S
✓ SECRET_M::SEQ_SECRET_SPEC [T= SECRET_M::SYSTEM_S_SEQ
✓ AUTH1_M::AuthenticateINITIATORToRESPONDERAgreement_na [T= AUTH1_M::SYSTEM_1

```
+.*
+.*
+.200 300 400 500 *
+.600 700 800 900 1,000 1,100 1,200 1,300 1,400 1,500 1,600 1,700 1,800
1,900 2,000 *
+.3,000 4,000 5,000 6,000 7,000 *
+.8,000 9,000 10,000 11,000 12,000 13,000 14,000 15,000 *
+.16,000 17,000 18,000 19,000 20,000 21,000 22,000 23,000 24,000 *
+.25,000 26,000 27,000 28,000 29,000 30,000 31,000 32,000 *
+.33,000 34,000 35,000 36,000 37,000 38,000 39,000 *
+.40,000 41,000 42,000 43,000 44,000 *
+.45,000 46,000 47,000 48,000 49,000 50,000 *
+.51,000 52,000 53,000 54,000 55,000 56,000 57,000 *
+.58,000 59,000 60,000 61,000 *
+.62,000 *
+.63,000 *
+.64,000 *
+.... 64,093
Refine checked 64,093 states
With 201441 transitions
Allocated a total of 44 pages of size 128K
Compaction produced 0 chunks of 16K.
Took 1(0+0) seconds
```

ALL_CRACKABLES
CHAOS(-)
CLOCK
CLOCK0(-)
DIRECT_MSG
ENV_MSG
INPUT_MSG
OUTPUT_MSG
SYSTEM
TIMEOUT
TOCKS(-)

FDR2 session: /home/yoney/CASPER_Related/Casper/PhDFiles/pros_with_ded_ind/Assumption_chp_spl/2CISSE06KKEWDD.csp

Figure 7.8: Achievement of intruder with deductions and induction on delayed decryption property

7.3.3 Attacker for the Proposed Timed Authentication Protocol and Frequent Key Renewal

Similar to the previous sections, the analysis and verification are done through the use of FDR and CSP respectively. The "#Protocol description" section of the designed protocol given in Section 5.3 is not modified, however the "#Intruder Information" section is modified with knowledge. Following script of the protocol shows the changes.

```
#Specification

StrongSecret(A, passwd(A,B), [B])
TimedAliveness(A, B,100)
TimedAgreement(A, B, 2, [kab])

#Intruder Information

Intruder = Mallory
IntruderKnowledge = {Alice, Bob, Mallory, Sam, Nm, Kab, PK, SPK, SK(Mallory), SKey(Mallory),\
passwd(Mallory,Alice), passwd(Mallory,Bob), \
passwd(Alice,Mallory), passwd(Bob,Mallory), passwd(Mallory,Mallory)}
Guessable = SessionKey
Crackable = SessionKey
Crackable = ServerKey
Crackable = Password
```

`IntruderKnowledge = {..., Kab,.., \ }`
`Guessable = SessionKey` and `Crackable = SessionKey`
are the entities added to the intruder's knowledge.

The CSP models of the legitimate principals and the communication channels with message transformations were generated using CASPER. The intruder model with deductions and inductions, the feasibility of the protocol and the achievements of *intruder* with deductive knowledge are demonstrated.

Entities that are initially known to intruder with *Guessable* and *Crackable* values are represented as follows:

```
IK0_init = union({Alice, Bob, Mallory, Sam, Nm, Km, SK(Mallory), SKey(Mallory),
            passwd(Mallory, Alice), passwd(Mallory, Bob),
            passwd(Alice, Mallory), passwd(Bob, Mallory),
            passwd(Mallory, Mallory), Garbage}, TimeStamp)
```

```
Guessable0 = Password
Guessable = diff(Guessable0, IK1)
Crackable0 = SessionKey
Crackable = diff(Crackable0, IK1)
```

Referring to the following script, the intruder closes up its initial knowledge under deductions and calculates the *facts* that can not be learnt.

```
components_(Sq.ms_) =
    if member(Sq.ms_, Fact_1) then {Sq.ms_} else set(ms_)
  components_(m_) = {m_}

  Seeable_ =
    Union({unknown_(components_(m_)) | (_,m_,_,_) <- SYSTEM_M::INT_MSG_INFO})

  Close_(IK_, ded_, fact_) =
    let IK1_ =
          union(IK_, {f_ | (f_,l_,fs_) <- ded_, fs_ <= IK_})
        ded1_ =
          {(f_,l_,fs_) | (f_,l_,fs_) <- ded_, fs_ <= fact_}
        fact1_ = Union({IK_, {f_ | (f_,l_,fs_) <- ded_},
                     Seeable_, Guessable0})
    within
    if card(IK_)==card(IK1_) and card(ded_)==card(ded1_)
       and card(fact_)==card(fact1_)
    then (IK_, {(f_,l_,fs_) | (f_,l_,fs_) <- ded_, not(fs_ <= IK_)}, fact_)
    else Close_(IK1_, ded1_, fact1_)
```

Components of the intruder are grouped as *currently unknown* ***fact*** *f_:* , *currently unknown* ***fact*** *f_, after a* ***guess****:*, *known* ***fact*** *f_:* and *known* ***fact*** *f_ after* ***guess****:*.
To put it clearly, *currently unknown* ***fact*** *f_:* is verified with a function called IGNORANT since the intruder tries to guess any **facts** with possible deductions.

```
 IGNORANT(f_,ms_,fss_,ds_) =
    hear?m_:ms_ -> KNOWS(f_,ms_,fss_,ds_)
    []
    ([] (l_, fs_) : fss_, not(member(f_,fs_)) @
        infer.(f_,l_,fs_) -> KNOWS(f_,ms_,fss_,ds_))
    []
    (member(f_,Guessable) & guess.f_ -> KNOWS''(f_,ms_,fss_,ds_))
    []
    guess?g_:diff(Guessable,{f_}) -> IGNORANT'(f_,ms_,fss_,ds_)
    []
    member(f_, ALL_CRACKABLES) & crack.f_ -> KNOWS(f_,ms_,fss_,ds_)
```

As the execution of the protocol steps continue, the intruder continues with the other components. The following script represents the intruder *currently unknown* ***fact*** *f_, after a*

guess:. The function `IGNORANT'` represents *currently unknown* ***fact*** *f_*, where `then([]g_ : Guessable @ verify.(f_,g_) -> vsync.g_ -> KNOWS(f_,ms_,fss_,ds_))` represents the *currently unknown* ***fact*** *f_, after a* ***guess:***

```
IGNORANT'(f_,ms_,fss_,ds_) =
    ([] (l_,fs_) : fss_, not(member(f_,fs_)) @
      infer.(f_,l_,fs_) ->
      if member(f_,ASYMMETRIC_KEYS) and member(inverse(f_), KnowableFact)
      then ([] g_ : Guessable @
              verify.(f_,g_) -> vsync.g_ -> KNOWS(f_,ms_,fss_,ds_))
           []
           notVerify.f_ -> KNOWS'(f_,ms_,fss_,ds_)
      else KNOWS'(f_,ms_,fss_,ds_)
    )
    []
    vsync?g_:diff(Guessable,{f_}) -> IGNORANT(f_,ms_,fss_,ds_)
```

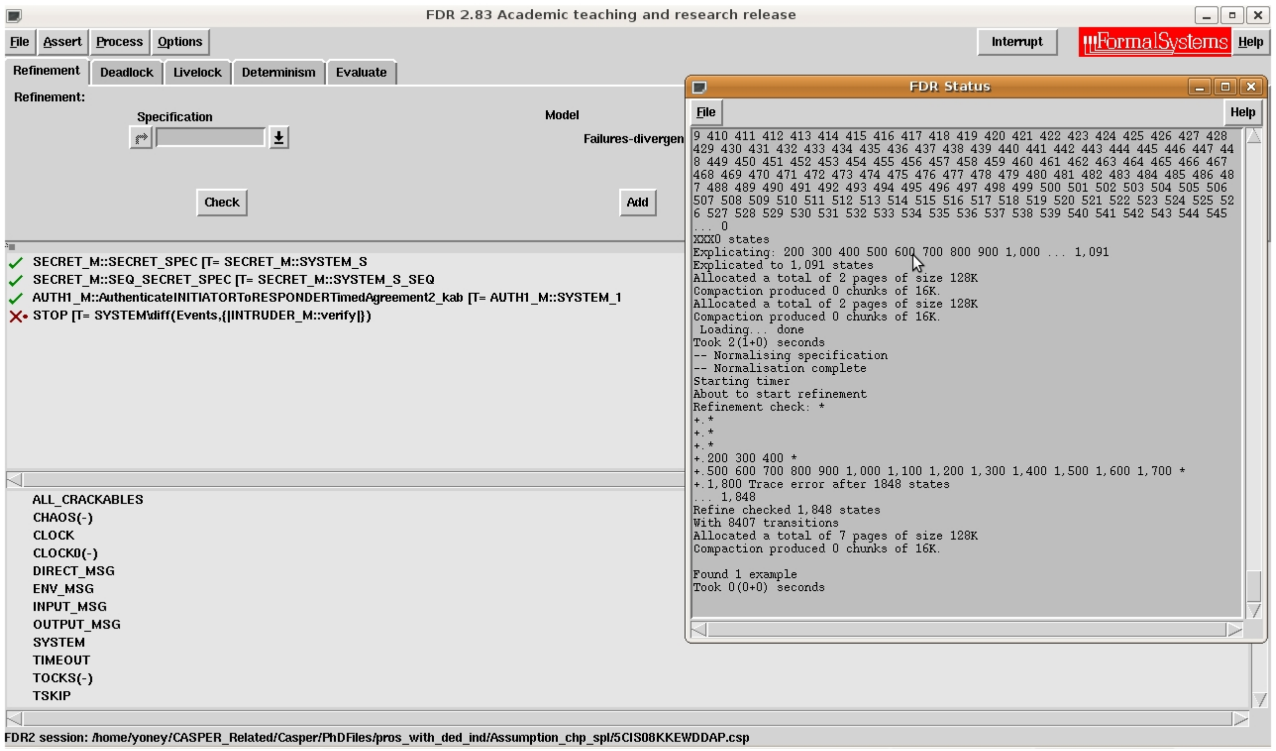
The function `KNOWS` represents *known* ***fact*** *f_:* to the intruder under possible deductions since `leak.f_ -> KNOWS(f_,ms_,fss_,ds_)` represents possible currently known entities that intruder guessed.

```
KNOWS(f_,ms_,fss_,ds_) =
    hear?m_:ms_ -> KNOWS(f_,ms_,fss_,ds_)
    []
    say?m_:ms_ -> KNOWS(f_,ms_,fss_,ds_)
    []
    ([] ded@@(f1_,l_,fs_) : ds_, f1_!=f_ @ infer.ded -> KNOWS(f_,ms_,fss_,ds_))
    []
    member(f_,ALL_SECRETS_DI) & leak.f_ -> KNOWS(f_,ms_,fss_,ds_)
    []
    ([] (l_,fs_) : fss_ @ infer.(f_,l_,fs_) -> KNOWS(f_,ms_,fss_,ds_))
    []
    guess?g_ -> KNOWS'(f_,ms_,fss_,ds_)
    []
    member(f_, ALL_CRACKABLES) & crack.f_ -> KNOWS(f_,ms_,fss_,ds_)
```

Last but not the least component of the intruder is *known* ***fact*** *f_ after* ***guess:***. It is clearly rendered that the function `KNOWS'` represents *known* ***fact*** *f_* and the
`vsync?g_:diff(Guessable,{f_})-> KNOWS(f_,ms_,fss_,ds_)`
which comes after the `infer.(f_,l_,fs_) -> [] g_ : Guessable @ verify.(f_,g_)->` shows the intruder knowledge with a possible guess for a *known* ***fact*** *f_* .

```
KNOWS'(f_,ms_,fss_,ds_) =
    infer?(f1_,l_,fs_) : ds_ -> KNOWS'(f_,ms_,fss_,ds_)
    []
    ([] (l_,fs_) : fss_ @
        infer.(f_,l_,fs_) -> [] g_ : Guessable @ verify.(f_,g_) ->
        vsync.g_ -> KNOWS(f_,ms_,fss_,ds_))
    []
    vsync?g_:diff(Guessable,{f_}) -> KNOWS(f_,ms_,fss_,ds_)
    []
    member(f_,ASYMMETRIC_KEYS) and member(inverse(f_), KnowableFact) &
      ([] g_ : Guessable @
         verify.(inverse(f_),g_) -> vsync.g_ -> KNOWS(f_,ms_,fss_,ds_))
```

Referring to Figure 7.9, power of the intruder with deductive knowledge on each specification of the protocol is analysed. Note that in the outline of the FDR results that follow the intruder named *Mallory*, while the legitimate agents INITIATOR and RESPONDER are *Alice* and *Bob* respectively. During the analysis where assertions failed, the process tree, $SYSTEM\_1$ of the FDR **debugger** were checked. As it is expressed in Sections 4.3 and 5.3, the failed assertions correspond to instances of attacks. The analysis performed is presented under deductions followed by deductions allocated with inductions.



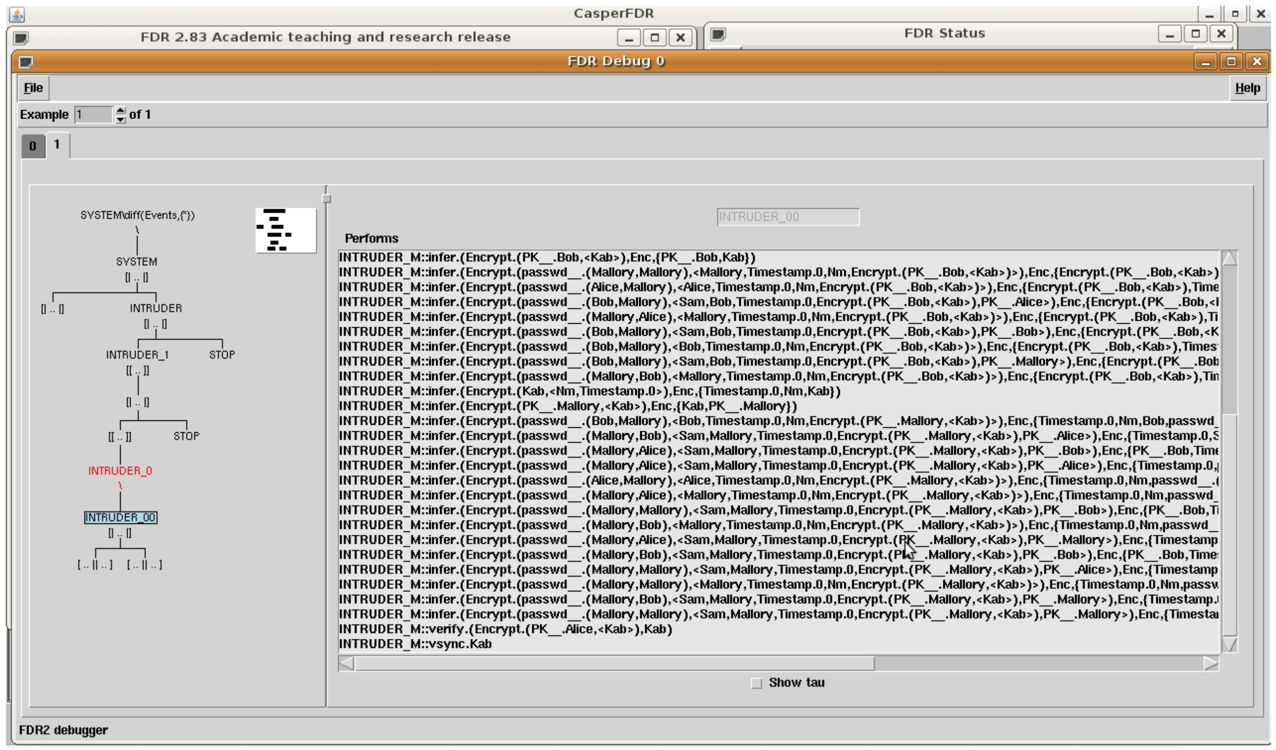
```
✓ SECRET_M::SECRET_SPEC [T= SECRET_M::SYSTEM_S
✓ SECRET_M::SEQ_SECRET_SPEC [T= SECRET_M::SYSTEM_S_SEQ
✓ AUTH1_M::AuthenticateINITIATORToRESPONDERTimedAgreement2_kab [T= AUTH1_M::SYSTEM_1
✗ STOP [T= SYSTEM\diff(Events,{|INTRUDER_M::verify|})
```

```
ALL_CRACKABLES
CHAOS(-)
CLOCK
CLOCK0(-)
DIRECT_MSG
ENV_MSG
INPUT_MSG
OUTPUT_MSG
SYSTEM
TIMEOUT
TOCKS(-)
TSKIP
```

```
9 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428
429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 44
8 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467
468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 48
7 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506
507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 52
6 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545
... 0
XXX0 states
Explicating: 200 300 400 500 600 700 800 900 1,000 ... 1,091
Explicated to 1,091 states
Allocated a total of 2 pages of size 128K
Compaction produced 0 chunks of 16K.
Allocated a total of 2 pages of size 128K
Compaction produced 0 chunks of 16K.
 Loading... done
Took 2(1+0) seconds
-- Normalising specification
-- Normalisation complete
Starting timer
About to start refinement
Refinement check: *
+.*
+.*
+.*
+.200 300 400 *
+.500 600 700 800 900 1,000 1,100 1,200 1,300 1,400 1,500 1,600 1,700 *
+.1,800 Trace error after 1848 states
... 1,848
Refine checked 1,848 states
With 8407 transitions
Allocated a total of 7 pages of size 128K
Compaction produced 0 chunks of 16K.

Found 1 example
Took 0(0+0) seconds
```

FDR2 session: /home/yoney/CASPER_Related/Casper/PhDFiles/pros_with_ded_ind/Assumption_chp_spl/5CIS08KKEWDDAP.csp

(a) Intruder with deductions and inductions



```
INTRUDER_M::infer.(Encrypt.(PK__.Bob,<Kab>),Enc,{PK__.Bob,Kab})
INTRUDER_M::infer.(Encrypt.(passwd__.(Mallory,Mallory),<Mallory,Timestamp.0,Nm,Encrypt.(PK__.Bob,<Kab>)>),Enc,{Encrypt.(PK__.Bob,<Kab>)
INTRUDER_M::infer.(Encrypt.(passwd__.(Alice,Mallory),<Alice,Timestamp.0,Nm,Encrypt.(PK__.Bob,<Kab>)>),Enc,{Encrypt.(PK__.Bob,<Kab>),Time
INTRUDER_M::infer.(Encrypt.(passwd__.(Bob,Mallory),<Sam,Bob,Timestamp.0,Encrypt.(PK__.Bob,<Kab>),PK__.Alice>),Enc,{Encrypt.(PK__.Bob,<I
INTRUDER_M::infer.(Encrypt.(passwd__.(Mallory,Alice),<Mallory,Timestamp.0,Nm,Encrypt.(PK__.Bob,<Kab>)>),Enc,{Encrypt.(PK__.Bob,<Kab>),Ti
INTRUDER_M::infer.(Encrypt.(passwd__.(Bob,Mallory),<Sam,Bob,Timestamp.0,Encrypt.(PK__.Bob,<Kab>),PK__.Bob>),Enc,{Encrypt.(PK__.Bob,<K
INTRUDER_M::infer.(Encrypt.(passwd__.(Bob,Mallory),<Bob,Timestamp.0,Nm,Encrypt.(PK__.Bob,<Kab>)>),Enc,{Encrypt.(PK__.Bob,<Kab>),Times
INTRUDER_M::infer.(Encrypt.(passwd__.(Bob,Mallory),<Sam,Bob,Timestamp.0,Encrypt.(PK__.Bob,<Kab>),PK__.Mallory>),Enc,{Encrypt.(PK__.Bob
INTRUDER_M::infer.(Encrypt.(passwd__.(Mallory,Bob),<Mallory,Timestamp.0,Nm,Encrypt.(PK__.Bob,<Kab>)>),Enc,{Encrypt.(PK__.Bob,<Kab>),Tin
INTRUDER_M::infer.(Encrypt.(Kab,<Nm,Timestamp.0>),Enc,{Timestamp.0,Nm,Kab})
INTRUDER_M::infer.(Encrypt.(PK__.Mallory,<Kab>),Enc,{Kab,PK__.Mallory})
INTRUDER_M::infer.(Encrypt.(passwd__.(Bob,Mallory),<Bob,Timestamp.0,Nm,Encrypt.(PK__.Mallory,<Kab>)>),Enc,{Timestamp.0,Nm,Bob,passwd_
INTRUDER_M::infer.(Encrypt.(passwd__.(Mallory,Bob),<Sam,Mallory,Timestamp.0,Encrypt.(PK__.Mallory,<Kab>),PK__.Alice>),Enc,{Timestamp.0,S
INTRUDER_M::infer.(Encrypt.(passwd__.(Mallory,Alice),<Sam,Mallory,Timestamp.0,Encrypt.(PK__.Mallory,<Kab>),PK__.Bob>),Enc,{PK__.Bob,Time
INTRUDER_M::infer.(Encrypt.(passwd__.(Mallory,Alice),<Sam,Mallory,Timestamp.0,Encrypt.(PK__.Mallory,<Kab>),PK__.Alice>),Enc,{Timestamp.0,
INTRUDER_M::infer.(Encrypt.(passwd__.(Alice,Mallory),<Alice,Timestamp.0,Nm,Encrypt.(PK__.Mallory,<Kab>)>),Enc,{Timestamp.0,Nm,passwd__.(
INTRUDER_M::infer.(Encrypt.(passwd__.(Mallory,Alice),<Mallory,Timestamp.0,Nm,Encrypt.(PK__.Mallory,<Kab>)>),Enc,{Timestamp.0,Nm,passwd_
INTRUDER_M::infer.(Encrypt.(passwd__.(Mallory,Mallory),<Sam,Mallory,Timestamp.0,Encrypt.(PK__.Mallory,<Kab>),PK__.Bob>),Enc,{PK__.Bob,Ti
INTRUDER_M::infer.(Encrypt.(passwd__.(Mallory,Bob),<Mallory,Timestamp.0,Nm,Encrypt.(PK__.Mallory,<Kab>)>),Enc,{Timestamp.0,Nm,passwd_
INTRUDER_M::infer.(Encrypt.(passwd__.(Mallory,Alice),<Sam,Mallory,Timestamp.0,Encrypt.(PK__.Mallory,<Kab>),PK__.Mallory>),Enc,{Timestamp
INTRUDER_M::infer.(Encrypt.(passwd__.(Mallory,Bob),<Sam,Mallory,Timestamp.0,Encrypt.(PK__.Mallory,<Kab>),PK__.Bob>),Enc,{PK__.Bob,Time
INTRUDER_M::infer.(Encrypt.(passwd__.(Mallory,Mallory),<Sam,Mallory,Timestamp.0,Encrypt.(PK__.Mallory,<Kab>),PK__.Alice>),Enc,{Timestamp
INTRUDER_M::infer.(Encrypt.(passwd__.(Mallory,Mallory),<Mallory,Timestamp.0,Nm,Encrypt.(PK__.Mallory,<Kab>)>),Enc,{Timestamp.0,Nm,passw
INTRUDER_M::infer.(Encrypt.(passwd__.(Mallory,Bob),<Sam,Mallory,Timestamp.0,Encrypt.(PK__.Mallory,<Kab>),PK__.Mallory>),Enc,{Timestamp.
INTRUDER_M::infer.(Encrypt.(passwd__.(Mallory,Mallory),<Sam,Mallory,Timestamp.0,Encrypt.(PK__.Mallory,<Kab>),PK__.Mallory>),Enc,{Timesta
INTRUDER_M::verify.(Encrypt.(PK__.Alice,<Kab>),Kab)
INTRUDER_M::vsync.Kab
```

(b) Power of intruder on SYSTEM for authentication specification

Figure 7.9: Achievement of the intruder with deductions and induction on frequent key renewal

7.4 Conclusion

Constructing attacker model with inductive and deductive approaches for the designed protocols presented in Sections 3.2, 4.2 and 5.2 is the main concern of this chapter. In Section 7.2, attacker analysis, formal methods for verifying security properties of cryptographic systems, that analyses attacker potentials [Lowe, 1996], [Paulson, 1998], [Roscoe *et al.*, 2009] are discussed. New attackers are modelled.

The capability of the attacker under assumptions of deductive and inductive approach are presented with failure refinement capability of FDR in Section 7.3. The CSP models of the legitimate principals and the communication channels with message transformations were generated using CASPER. The intruder model with deductions, the feasibility of the protocol and the achievements of the *intruder* with deductive knowledge are demonstrated. Thus, addition of induction confirms a fresh attack against the agents (of the protocols presented in Sections 3.2, 4.2 and 5.2), which was not possible with intruder ability defined under possible deductions only. Secondly, it is observed that the addition of induction has significantly reduced the number of steps registered for the respective steps.

Chapter 8

Conclusion

8.1 Introduction

This thesis describes original research on development of security strategies using Kerberos in wireless communication networks. Due to the critical nature of wireless communication networks, the existing methods are insufficient to address perspective and requirements in authentication design. Since Kerberos authentication protocol is considered as a solution, various existing methods and solution techniques for Kerberos, its basic operation in wireless communication networks are described, studied and analysed. New authentication protocols are proposed. As an effective tool in protocol analysis, CASPER is extensively used. In order to verify the accuracy of the developed protocols, formal verification methods are employed. Results are presented in order to show the effectiveness of the protocols proposed. As an effective verification tool, CSP is extensively used for this purpose. These methods are helpful for understanding the complex interaction between authentication protocols' entities and the network concerned. It is expected that, increased security resulted in degradation in system performance. It is essential to ensure that any degradation in system performance is at acceptable levels. For this purpose, analytical models have been developed to evaluate the performance of the systems considered.

8.2 Contributions of the Thesis

The major contributions of the thesis can be summarised as follows:

1. In Chapter Three, a framework is presented to provide a variant for Kerberos security protocol for IEEE 802.11 wireless LANs that require a high level of security.

The framework is extended as the provisions of [Eneh *et al.*, 2004]. Framework's paradigms and entities are finalised according to trusted third party authentication protocol features. The legitimate entities of a particular wireless LAN environment, KDC and TGS of Kerberos adopt the challenge-response paradigm and the program and data containing the credentials such as the identities of the devices (such as MAC addresses) are installed on each of the entities as well as TGS and KDC.

In the second part of Chapter Three, the framework and its expression of particular security properties are designed within CSP that provides a foundation for verification. Additionally, in terms of authentication and authorisation, security aspects of the protocol's availability is checked with CASPER. The theoretical grounds of a commonly used protocol, Kerberos, its implications and the capability of the attacker under assumptions of possible deductions are presented with inductive capability in CASPER/FDR. This protocol is successful as initial steps for the proposed model. It has been verified that the framework minimises the attacks known to the literature.

2. Depending on the proposed framework presented in Chapter Three new improvements are introduced on the specifications and description of the proposed protocol in Chapter Four for increasing the decryption time to reduce the likelyhood of success of intruders' attacks. Kerberos and Key-Exchange authentication protocols are combined and the design of this new protocol depends on the *delay decryption* property of Kerberos [Neuman and Ts'o, 1994]. Protocol description is designed and featured by considering the changes in the EAP structure [Marin-Lopez *et al.*, 2009], [Zrelli and Shinoda, 2007] and Kerberos assisted authentication in wireless communication networks [Pirzada and McDonald, 2004].

   Critical analysis to find out errors associated with this protocol for unassailable attacks and demonstration of the feasibility of the protocol and the achievements of *intruder* with certain knowledge are considered in the second part of this chapter. The proposed protocol is checked through CASPER and FDR. Due to the *delay decryption* introduced, there were "*no attack found*". The ascertains are checked to confirm the design is an exact refinement. The check is successful, when there is no possible attacks against the protocol. In the CSP representation, the protocol is modelled as a network and specified the authentication property. Advantage of the extensibility of CSP gives the opportunity to add additional elements to the protocol's entities. Since the proposed protocol is time sensitive (i.e. timed authentication protocol), introduction of any delay prevented any intruder's attempt to launch an attack. Because of this, the *delay decryption* technique used in this chapter has proven to delay intruders trying to break encryption.

3. In Chapter Five the designed and verified protocols of Chapter Three and Four, are extended to increase the breaking time of encryption for an intruder. A new approach is derived by restricting access to the authentication system and during this time, distribution of the frequently renewed keys take place, still the encryption/decryption of messages are controlled with timed authentication. The authentication protocol adopted the challenge-response paradigm defined in Section 3.2 and the aforementioned protocol description with *delay decryption* of Section 4.2. When the protocol is checked through FDR, due to the *restricted access* during *frequent key renewal* there were "*no attack found*". In order to find the capability of the attacker under assumptions, improvements are made on the entities' encryption elements. Results are compared with inductive capability of FDR.

   In the rest of the Chapter Five, the CSP codes and the construction of rank functions of the proposed protocol are presented. CSP processes and rank functions that are constructed, give opportunities of better understanding of security protocols and focuses on relevant design aspects of these protocols. CSP processes and rank function theorem [Schneider, 1998], [Roscoe, 1995] are applied to expose any flaws in the design and possible attacks have been successfully identified. Analysis shows that the protocol developed has achieved the goal of increasing the time needed to break the encryption, and hence improve security. Since the proposed approach involves temporary interruption to link/server access, it has implications in terms of performance degradation.

4. In Chapter Six, performability evaluation approaches are presented for the protocols proposed in Chapter Five. The proposed approach has implications in the performance of the underlying network. Although security is increased, it is expected that any degradation in system performance should be at acceptable levels. For this purpose, a mathematical model is developed in Section 6.2 to evaluate the performance of the system considered and the MQLs are calculated using various values for important system parameters. Performance degradation of the system has been evaluated for

various key distribution times while access to the authentication server is restricted at certain intervals. Results enable designers to carefully choose parameters for the best performance results for acceptable levels of security. Section 6.3 is concerned with a modelling approach for performability evaluation of Kerberos servers which dynamically renew keys under pseudo-secure conditions. Numerical results are obtained and presented for various performability measures for different key renewal and service interruption period values. Unlike the previous studies, the server failures are considered as well. The approach presented in this chapter provides more realistic performability measures. Results showed that the server failures, as well as key renewal times and time between interruptions can significantly affect the performance of a Kerberos server especially if high arrival rates are expected. The model developed is highly flexible and it can be used for systems with various failure, repair, and renewal times and times between interruptions. The method can be extended for multiple Kerberos servers and for systems with backup servers especially for the KDC.

5. In Chapter Seven, constructing an attacker model with inductive and deductive approaches for the proposed protocols presented in Sections 3.2, 4.2 and 5.2 is discussed. In Section 7.2, attacker analysis, formal methods for verifying security properties of cryptographic systems that analyses attacker potentials are discussed and new attackers are modelled.

   In Section 7.2 attacker definition is built into the CSP models of proposed protocols given in Sections 3.2, 4.2, 5.2 and 5.3, and subsequently tested using FDR. Section 7.3, tests proposed protocols by comparing *protocol description* and *specification.* When the protocol investigated against to attack, *protocol description* fails to be a refinement of the *specification.* In terms of authentication, this section is concerned with increasing power of an intruder to break an encryption with inductions and deductions. Results of times of breaking the encryption for the proposed protocols and intruder's power are presented and discussed. Referring to Section 7.3 power of an intruder with deductive knowledge on each specification of the protocol is analysed. During the analysis, the failed assertions correspond to instances of attacks. The analysis performed is presented under deductions followed by deductions combined with inductions.

To render a conclusion, the contributions of this thesis are various authentication models designed for Kerberos in wireless communication systems, the validation of these models by formal verification tools in order to make provision for the formalisation of these models and also, extend the intruder abilities by adding those of possible induction to existing deduction in CSP. Essentially, this research attempts to combine the inductive and deductive approaches in model checking in order to realise an abstract description of intruder with enhanced capabilities. Results of verification of protocols using the CSP/FDR approach, with intruder capability defined as an interleaved process of induction and deduction, show that protocols are subject to attacks not previously recorded. Additionally, this research aims to derive effects of these models on performance degradation as well as availability of Kerberos. This is provided by designing analytical models and empirical results are derived and discussed.

8.3 Suggestions for Future Study

The work done paths the way for the further development work. Some suggestions for future study are given below

1. Although security is increased with different authentication properties for Kerberos, such as *delay decryption*, it is expected that any degradation in system performance should be at acceptable levels. For this purpose, mathematical models are developed in Sections 6.2 and 6.3 to evaluate the performance of the system considered. Unlike the previous studies, the server failures are considered as well for the model developed in Section 6.3. The model developed is highly flexible and it could be used for systems with various failure, repair, and renewal times and times between interruptions. The method can be extended for multiple Kerberos servers and for systems with backup servers especially for the KDC for improved reliablility.

2. In Chapter Seven, attacker potentials are discussed and new attackers are modelled. In case of using attacker models based on combined deductive and inductive methods, protocol analysis carried out using the FDR, shows increased ability in attackers to succeed.

   It is possible to develop a CSP description of an operator induced by an inductive definition and uses this for the development of the capability of an attacker. During the analysis, this would show to what , the attacker can deduce and induce facts from sets of messages that it sees.

3. The architectures provided in Chapters Four and Five can be further extended for highly available, highly reliable wireless communication systems. It is possible to use more than one Kerberos servers with appropriate modifications to provided better performance, reliability and performability. Hot, and cold standby backup strategies, as well as various multi server architectures are available in the literature for similar applications.

Bibliography

M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148:36–47, 1999.

M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Trans. Softw. Eng.*, 22(1):6–15, 1996.

B. Aboba and P. Calhoun. RADIUS (remote authentication dial in user service) support for extensible authentication protocol (EAP). RFC, The Internet Society, United States, 2003.

N. Baghaei and R. Hunt. Security performance of loaded IEEE 802.11b wireless networks. *Computer Communications*, 27(17):1746–1756, 2004.

J. Banks, J. S. Carson, B. L. Nelson, and D.M. Nicol. *Discrete-Event System Simulation.* Prentice-Hall: Upper Saddle River, NJ., 4th. edition, 2005.

G. Bella. *Inductive Verification of Cryptographic Protocols.* PhD thesis, Clare College, University of Cambridge, March 2000.

S. M. Bellovin and M. Merritt. Limitations of the kerberos protocol. In *Winter 1991 USENIX Conference Proceedings, USENIX Association*, 253–267, 1991.

V. A. Brennen. Kerberos infrastructure HOWTO. Technical report, CryptNET, Guerrilla Technology Development, 2004.

M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *SIGOPS Oper. Syst. Rev.*, 23(5):1–13, 1989. ISSN 0163-5980.

R. Chakka and I. Mitrani. Heterogeneous multiprocessor systems with breakdowns: Performance and optimal repair strategies. *Theor. Comput. Sci.*, 125(1):91–109, 1994.

P. Congdon, B. Aboba, A. Smith, G. Zorn, and J. Roese. IEEE 802.1x remote authentication dial in user service (RADIUS) usage guidelines. RFC, The Internet Society, United States, 2003.

P. Cousot and R. Cousot. Inductive definitions, semantics and abstract interpretation. In *POPL*, 83–94, 1992.

C. Cremers and P. Lafourcade. Comparing state spaces in automatic protocol verification. In *Proc. of the Seventh International Workshop on Automated Verification of Critical Systems (AVoCS'07)*, Electronic Notes in Theoretical Computer Science. Elsevier ScienceDirect, September 2007.

C. J. F. Cremers, P. Lafourcade, and P. Nadeau. Comparing state spaces in automatic protocol analysis. In *Formal to Practical Security*, vol. 5458/2009 of *Lecture Notes in Computer Science*, 70–94. Springer Berlin / Heidelberg, 2009.

W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, Nov 1976.

D. Dolev and A. C. Yao. On the security of public key protocols. In *SFCS '81: Proceedings of the 22nd Annual Symposium on Foundations of Computer Science*, 350–357, Washington, DC, USA, 1981. IEEE Computer Society.

B. Dutertre and S. Schneider. Using a PVS embedding of CSP to verify authentication protocols. In *Proceedings of the 10th International Conference on Theorem Proving in Higher Order Logics*, TPHOLs '97, 121 –136, London, UK, 1997. Springer-Verlag. ISBN 3-540-63379-0.

A. Eneh, H. Singh, and O. Gemikonakli. Three way authentication framework for IEEE 802.11b networks. In *Proc. of the 4th Int. Net. Conf. INC '04*, 2004.

A. Eneh, O. Gemikonakli, and R. Comley. Security of electronic commerce authentication protocols in economically deprived communities. In *The 5th Security Conference*, ISBN: 0–9772107–2–3, Las Vegas, Nevada, April 2006.

E. Ever. *Performability Modelling of Homogeneous and Heterogeneous Multi-server Systems with Breakdowns and Repairs*. PhD thesis, School of Computing Science, Middlesex University, November 2007.

J. Geier. Minimising wireless local area network security threats, 2005. URL `http://www.wi-fiplanet.com/tutorials/articles.php/1457211[Accessed:August2010]`.

A. Harbitter and D. A. Menascé. A methodology for analyzing the performance of authentication protocols. *ACM Trans. Inf. Syst. Secur.*, 5(4):458–491, 2002.

B. R. Haverkort and A. Ost. Steady-state analysis of infinite stochastic petri nets: Comparing the spectral expansion and the matrix-geometric method. In *Proceedings of the 6th International Workshop on Petri Nets and Performance Models*, page 36, Washington, DC, USA, 1997. IEEE Computer Society. ISBN 0-8186-7931-X.

C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21: 666–677, 1985.

R. Jain. *The Art of Computer Systems Performance Analysis.* John Wiley & Sons, Inc., 1991.

Y. Jiang, C. Lin, X. Shen, and M. Shi. Mutual authentication and key exchange protocols with anonymity property for roaming services. In *NETWORKING*, 114–125, 2005.

M. A. Kâafar, L. B. Azzouz, F. Kamoun, and D. Males. A kerberos-based authentication architecture for wireless lans. In *NETWORKING*, 1344–1353, 2004.

A. Kahate. *Cryptography and Network Security.* Tata McGraw-Hill: New Delhi, 2nd edition, 2009.

A. Kehne, J. Schönwälder, and H. Langendörfer. A nonce-based protocol for multiple authentications. *SIGOPS Oper. Syst. Rev.*, 26(4):84–89, 1992.

J. Klensin, R. Catoe, and P. Krumviede. IMAP/POP authorize extension for simple challenge/response. RFC, The Internet Society, United States, 1997.

J. Kohl and C. Neuman. The kerberos network authentication service (v5). RFC, MIT/ The Internet Society, United States, 1993.

A. Law and W.D. Kelton. *Simulation Modelling and Analysis.* McGraw-Hill: NY, USA., 3rd edition, 2000.

G. Lowe. An attack on the needham-schroeder public-key authentication protocol. *Information Processing Letters*, 56(3)(3):131–133, 1995.

G. Lowe. Some new attacks upon security protocols. In *9th IEEE Computer Security Workshops*, 162–169. Society Press, 1996.

G. Lowe. Casper: A compiler for the analysis of security protocols. In *10th Computer Security Foundations Workshop (CSFW '97)*, 18–30. IEEE Computer Society, June 10-12 1997.

G. Lowe. A hierarchy of authentication specification. In *10th Computer Security Foundations Workshop (CSFW '97), June 10-12, 1997, Rockport, Massachusetts, USA*, pages 31–44. IEEE Computer Society, 1997b.

G. Lowe, P. Broadfoot, C. Dilloway, and M. L. Hui. *Casper: A Compiler for the Analysis of Security Protocols*, 1.12 edition, September 2009.

R. Marin-Lopez, F. Pereniguez, Y. Ohba, F. Bernal, and A. F. G. Skarmeta. A transport-based architecture for fast re-authentication in wireless networks. In *SARNOFF'09: Proceedings of the 32nd international conference on Sarnoff symposium*, 40–44, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-3381-0.

C. A. Meadows. Formal verification of cryptographic protocols: A survey. In *Advances in Cryptology (ASIACRYPT'94)*, vol. 917 of *Lecture Notes in Computer Science*, 133–150. Springer Berlin / Heidelberg, 1995.

R. Milner. *Communicating and Mobile Systems: the Pi-Calculus.* Cambridge University Press, 1999.

A. Mishra and W. A. Arbaugh. An initial security analysis of the IEEE 802.1x standard. Technical report, UMIACS-TR-2002-10, 2002.

I. Mitrani. Approximate solutions for heavily loaded markov-modulated queues. *Perform. Eval.*, 62(1-4):117–131, 2005.

J. F. Monin. *Understanding Formal Methods.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001. ISBN 1852332476.

R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.

C. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications Magazine*, 32(9):33–38, September 1994.

M. S. Obaidat and N. A. Boudriga. *Fundamentals of Performance Evaluation of Computer and Telecommunications Systems.* Wiley-Interscience, New York, NY, USA, 2010. ISBN 0471269832.

L. C. Paulson. The inductive approach to verifying cryptographic protocols. *J. Comput. Secur.*, 6:85–128, January 1998.

A. A. Pirzada and C. McDonald. Kerberos assisted authentication in mobile ad-hoc networks. In *Proceedings of the 27th Australasian conference on Computer science (ACSC'04)*, 26, 41–46, 2004.

A. W. Roscoe. CSP and determinism in security modelling. In *In Proc. IEEE Symposium on Security and Privacy*, 114–127. Society Press, 1995.

A. W. Roscoe. *The Theory and Practice of Concurrency.* Prentice-Hall (Pearson), 2005.

A. W. Roscoe, P. J. Armstrong, and Pragyesh. Local search in model checking. In *Proceedings of the 7th International Symposium on Automated Technology for Verification and Analysis*, ATVA '09, 22–38, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-04760-2.

P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *The modelling and analysis of security protocols: the csp approach.* Addison-Wesley Professional, 2000.

S. Schneider. Verifying authentication protocols in csp. *IEEE Trans. Softw. Eng.*, 24(9): 741–758, 1998.

S. A. Schneider. *Concurrent and real time systems: the CSP Approach.* Addison-Wesley, 1999.

B. Schneier. *Applied Cryptography.* John Wiley and Sons, second edition, 1996.

SECWP. Security white paper evolution, requirements, and options. Technical report, Symbol Technologies Inc., 2007.

S. A. Shaikh and V. J. Bush. Analysing the WOO-LAM protocol using CSP and rank functions. *Journal of Research and Practice in Information Technology*, 38(1):19–29, 2006.

W. Stallings. *Cryptography and Network Security Principles and Practices.* New Jersey:Prentice Hall, 5th ed., 2010.

K. Trivedi and M. Xiaomin. Probabilistic analysis of wireless cellular networks. In *Proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, vol. 79, 27–44, 2002.

K. S. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications.* Wiley, NY, USA., 2002.

J. Vollbrecht, D. Rago, and R. Moskowitz. Wireless lan access control and authentication: 802.11b wireless networking and why it needs authentication. Technical report, Interlinks Networks Inc., 2001.

T. Y. C. Woo and S. S. Lam. A lesson on authentication protocol design. *SIGOPS Oper. Syst. Rev.*, 28(3):24–37, 1994.

Y. Zhao and N. Thomas. A cost model analysis of a secure key distribution centre. In *The 9th International Conference for Young Computer Scientists(ICYCS'08)*, 1969–1974, Washington, DC, USA, 2008. IEEE Computer Society.

Y. Zhao and N. Thomas. Efficient solutions of a pepa model of a key distribution centre. In Elaine Weyuker and Murray Woodside, editors, *Performance Evaluation*, vol. 67, 583–756. Elsevier Science Publishers B. V., 2009.

S. Zrelli and Y. Shinoda. Specifying kerberos over eap: Towards an integrated network access and kerberos single sign-on process. In *International Conference on Advanced Information Networking and Applications*, 490–497, Los Alamitos, CA, USA, 2007. IEEE Computer Society.